ACTIVITY IDENTIFICATION FROM ANIMAL GPS TRACKS WITH SPATIAL TEMPORAL CLUSTERING METHOD DDB-SMOT

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

SIMIAO SUN

Dr. Yi Shang, Advisor

April 2016

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

ACTIVITY IDENTIFICATION FROM ANIMAL GPS TRACKS
WITH SPATIAL TEMPORAL CLUSTERING METHOD DDB-SMOT

presented by Simiao Sun,
a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Yi Shang

Dr. Dong Xu

Dr. Timothy Trull

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Prof. Yi Shang of the Computer Science Department at University of Missouri - Columbia. The door to Prof. Yi office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it. And as the second reader of this thesis, I am gratefully indebted to him for his very valuable comments on this thesis.

I would also like to thank the experts who were involved in the data sharing for this research project: Dr. Joel Sartwell and Dr. Sherry Gao. Without their passionate participation and input, the thesis could not have been successfully completed.

Finally, I must express my very profound gratitude to my partners for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

ABSTRACT

With various GPS devices or services growing rapidly, large amount of GPS tracking data has been collected, both for human beings and wild animals. However, the raw GPS data cannot directly provide us with any valuable information because of the semantic gap between it and the raw GPS trajectory data. As a result, algorithms are needed to extract the semantic information from raw GPS data. To solve this problem, this project implements two software tools and a web application.

- **Semantic Analysis Software** provides semantic analysis based on stops in the trajectory detected by DDB-SMoT (Direction and Distance Based - Stop and Move of Trajectory) and POI (Point of Interest) list to output a list of activities in order to explain the meaning of the given trajectory.

- **Trajectory Generator Software** generates labeled trajectory based on the stop and move model to evaluate the performance of stops detection algorithms.

- **Semantic Analysis Web Application** displays the semantic enrichment process step by step on Google Map use bear and deer GPS trajectories provided by MDC (Missouri Department of Conservation).

Through experiments, the DDB-SMoT algorithm has an overall accuracy of 91.18% when detecting stops and movement points in animal trajectory generated by the trajectory generator. Because lack of a rich animal POI dataset and activity ground truth, the verification of the semantic analysis process will leave as future work for the project.

Chapter 1: Introduction

In the last few years, numerous wearable tracking devices to sense the movement of people, vehicles and even wild animals have generated large volumes of mobility data, representing the traces of people's and animal's activity. This project will mainly focus on analyzing animal activity with mobility data. Nowadays, lots of application areas could benefit from study of animal activity analysis such as wildlife population management, wildlife animal protection and even some scientific related researches. Although there are lots of data about animal movement and the accuracy of those devices are increasing, it still does not directly improve our understanding of meaning of animal GPS data. We currently do not have much information on how to fill up the semantic gap between the animal GPS trajectory data and the real semantic data. As a result of that, approaches are needed to automatically infer animals' activity given its location tracking. The method presented in this paper is to enrich animals movements, represented as GPS data, with semantic information about the activities performed during the time the GPS data provided. The main idea is to split the whole trajectory into moves and stops, assume that the purpose of the move for an animal is to move towards a place where it can perform some activity and the purpose of stop for an animal is to stay around a place to perform the activity. Under this assumption, there are two possible conditions for each GPS point in the trajectory, either it is moving towards the next interesting place to do something, or it stays around current place to perform an activity. Further, we want to predict, with a degree of approximation, what is the activity the animal is moving to. For example, a bear stopping around a river is fishing in the river, while when

staying around its own den is performing a resting or hibernating activity. In order to make a good prediction, the first and most crucial thing is to identify the places where animals have stopped as accurate as possible; secondly, we will try to associate each stop position with a most possible POI(Point of Interest); finally, with the most possible POI, we can use a mapping table to find the category that the POI belongs to and then map it to the proper activity performed around the place. This project implements this semantic enrichment procedure as software to take in GPS trajectory and POI dataset to identify the activities performed in the trajectory.

In current situation, to save the battery life, it gives out the signal around once an hour. Compared with human GPS trajectories, the black bear trajectory is really sparse. Therefore, it is necessary to find a proper way to find out accurate stops in the trajectory. Also, the accuracy of this approach in this paper depends largely on the richness of the POI data. However, Unlike the POI lists and categories for human semantic analysis are very easy to obtain, it is really hard to get the POI positions and categories for animals. So, the association between the stop and the POI should be defined properly in order to have a better result. Hence, there are mainly two core parts to improve the accuracy of the semantic enrichment algorithm. One way is to develop a high accuracy stop detection algorithm. Another way is to search for a rich POI list for the area of semantic enrichment process. Clearly, due to lack of animal POI data, there are not much we could do about it. Therefore, the stops detection algorithm can be modified to obtain a more accurate stops result. In this paper, we have developed the algorithm DDB-SMoT (Direction and Distance Based - Stop and Move of Trajectory) based on DB-SMoT (Direction Based - Stop and Move of Trajectory) presented in [6]. DB-SMoT algorithm only considers the direction

change threshold when filtering out movement points in a trajectory, which requires the trajectory to be very dense. However, with sparse trajectories like the bear data or deer data, distance change between consecutive points should also take into consideration. So, based on the model from DB-SMoT algorithm, more constraints are added in order to filter out movement points from stop clusters. With experiments produced by a wildlife trajectory generator presented in this paper, the DDB-SMoT shows better accuracy of finding stop and move points in the trajectory than the DB-SMoT algorithm.

In order to test the DDB-SMoT algorithm and find the proper parameters to run the algorithm, this project developed a wildlife trajectory generator to generate labeled trajectory that is similar to the real animal trajectory. The trajectory generator uses the stop and move model to generate stop or movement points and transits between those two model with a Markov probability model. It generates the next point in the trajectory based on a direction change and a distance change, which are generated with Gaussian random function with parameters learned from real animal trajectory. With features extracted from the different animal GPS tracks, it is able to monitor different kinds of animal trajectories. Also, the software provides with interfaces for a different stops detection algorithm to verify their accuracy on simulated animal trajectories.

This project also built a web application to visualize the results provided by the semantic analysis software. The semantic analysis software generates intermediate results like stop labels and position of stop centers for the website to display. The web application can provide users a platform to understand the semantic enrichment process. It displays all the intermediate result on the Google Map after each step in

the process to illustrate how the semantic analysis works. All the experiment data are from real black bear and deer GPS data provided by the Missouri Department of Conservation (MDC).

There are two phases of experiments in this project. First, the project tests the accuracy of DDB-SMoT algorithm and compares the results with DB-SMoT. With labeled animal trajectories generated from the trajectory generator, this new spatial temporal clustering method DDB-SMoT turned out to have 91.18% of overall accuracy for detecting stop points and move points in trajectory. It largely improved the accuracy of detecting move points in trajectory when compared with DB-SMoT algorithm. Second, the project tries to test the accuracy of the semantic enrichment process. However, without a rich POI dataset to produce the actual prediction of activity identification and ground truth that records the activity the bear or deer performed in the trajectory, there are no way to come up with an accuracy for the semantic analysis. So, in the future, the project will try to build up POI dataset according to locations provided by the stop detection result and observe activity performed by animals in order to verify the semantic enrichment process.

The paper introduces related works for both semantic analysis algorithms and spatial temporal clustering methods in Chapter 2. From Chapter 3 to Chapter 5, there are detailed explanation about the DDB-SMoT algorithm, semantic enrichment process and the trajectory generator. Experimented results and analysis are displayed in Chapter 6.

Chapter 2: Related Works

2.1 Related Works of Trajectory Semantic Analysis

There are very few studies on animal semantic analysis, however, there are actually a lot of works on human semantic modeling. The work in this paper is based on the approach introduced in work [1]. The author defines a semantic enrichment model for analyzing human GPS trajectories. The GPS tracks is defined as a spatial temporal function keeping record of the object moving in space during a given time interval, while the semantic information is defined as a set of stops and moves. Stops are the points that the object stays still at some place and performs an activity, while the moves are the points that the positions of the object changes. The basic assumption behind the concept of stop is that the place where a person or animal stops is of some interest for her/him. As a result of that, each stop could be related or associated to a POI(Point of Interest). With the POI information and probability model, an activity could be inferred from each stop.

The identification of mobile activity from GPS data of people is not a new field in data science. There are basically two types of research that are related. The first one is to identify the transportation means [15]. Using speed, acceleration and speed change rate, the authors first split the situation into two different condition: walk, non-walk. In the second step, they analyzed the non-walk segments into segments with transportation models: bicycle, bus, and driving. They used a combination of techniques,

from supervised learning to decision tree inference, and add a post-processing step to improve the accuracy of the segmentation. Another trend of research is to identify the human activity based on the trajectories. A similar approach to ours has been proposed in [12], here the authors present a method to automatically extract sequences of activities from a large set of trajectory data. The assumption is that activities may be carried out at a POI during a stop in the user trajectory. The function to map a stop representative to a valid POI is crucial and may depend on many factors. The first parameter that should be taken into consideration is the distance. The POIs that are far away from the stop should not be mapped to. Also, the duration of the stop, the time of the day and the date when the stop happened, they all are factors that may affect the meaning of the stop. They tested their algorithm using synthetically generated trajectories dataset with the POIs collected in a specific area in California. The drawback of this type testing is that there is no real validation of the method since there is no proof of the correctness of the inferred POIs.

The work of [4] is again in the direction of inferring activities from users trajectories. This paper presents an approach using spatial temporal attractiveness of POIs to identify activity-locations and durations from raw GPS trajectory. The algorithm they proposed finds the intersections of trajectories and spatial-temporal attractiveness prisms to indicate the potential possibilities for activities. The experiments use one month GPS trajectories from 10 volunteers where they show an high accuracy of the method. Though the approach using the Spatial Temporal POI Attractiveness(STPA) is a good way to start the semantic enrichment analysis, it is very hard to find a POI dataset that fits the STPA's need for human beings, not to mention, we need to do it for animals.

Our work is an implementation of the work [16]. In the work [16], the author purposed a gravity model to associate stops with POIs, and hence infer the activity during the stop. It does need a rich POI dataset to obtain a high accuracy of semantic enrichment prediction, therefore, we are still able to get a reasonable result with POIs that are easy to acquire.

Most of works are done for human activity inference, while there are very few works about wildlife behavior inference. This is probably because it is really difficult to get the ground truth and analyze the results of the present method. It is also difficult to continuously observe them, and it is even harder to understand the activity when we are trying to observe them and label the data. In our work, we focused on the implementation of the previous work, once we are able to obtain more accurate POI dataset, we get use them as an input to predict the wildlife animal activity through their trajectory.

2.2 Related Works of Spatial Temporal Clustering

The core part for every semantic analysis algorithm is to find the valuable points in the raw GPS trajectory to start the analysis. Since Spaccapietra [17] introduced a new model to reason about trajectories, which is called stops and moves. This model is especially interesting to add semantic information to raw trajectories. Generally speaking, stops are the key part of a trajectory from a semantic analysis point of view, while the moves are the travel the user or animals made in order to arrive at the stop place. This stop and move model can be used with a large number of applications, such as semantic analysis, traffic control, etc.

So far, there are three different methods that have been designed to detect stops in a trajectory: IB-SMoT(Interaction-Based Stop and Move of Trajectory) [18], CB-SMoT(Clustering-Based Stop and Move of Trajectory)[19] and DB-SMoT(Direction Based-Stop and Move of Trajectory)[6]. IB-SMoT generates stops and moves based on the intersections between GPS points in the sample trajectory and the list of given geographic object types that are specially interesting to the specific application. This intersection requires a minimum time threshold for the sub-trajectory be considered as a stop. This algorithm is used by the application to detect for short amount of time duration for interesting places, such as tourism.

CB-SMoT(Clustering-Based Stop and Move of Trajectory) is a clustering method based on the speed variation of the trajectory. It directly measures the trajectory of the GPS points and generates clusters in places where the speed of the sub-trajectory is lower than a given limit of the speed threshold for a minimal amount of time. As a step to revise the result in the first step, the method matches the stop clusters with a set of predefined relevant geographic places that are interesting to the application. This method is particularly useful for applications in which the speed plays an significant role, such as traffic management system.

DB-SMoT(Direction-Based Stop and Move of Trajectory) is a spatial temporal clustering method based on the direction change between each of two consecutive points of the trajectory. It measures the direction change respect to a minimum direction change with threshold and cluster stop points in places where the direction change is larger than the threshold. The assumption for this is that when the user is moving towards somewhere, the direction change is much lower, on the other hand, when the user stops at some places, it moves around the place which makes the

direction change between points become much larger. This algorithm is very useful for applications that need to find the stops with long time duration.

In our case, we use GPS trajectory collected from the wildlife animal. In order to battery, the GPS device cannot send out frequent signal, instead it sends out the GPS position once an hour. Therefore, we do not get a continuous trajectory. Unlike human beings, wild animal always moves in different areas, there are no relatively fixed areas they would live in. As a result of that, it is almost impossible for us to obtain detailed geographic object types or information for such a wide area. In this case, the two spatial temporal clustering method explained above are not a good fit to our project. Both IB-SMoT and CB-SMoT method require too much geographic information and only used for stops around known places. In our case, it is really hard to human beings to obtain the geographic information that is not inside a city, and we don't have existing dataset about where the wildlife animals should stop. On the contrary, those information that are needed for implementing IB-SMoT is exactly what we should analyze from the raw trajectory. As a result of that, IB-SMoT is not a good fit to our project.

For this paper, we found the DB-SMoT algorithm is a better fit. It does not require any geographic information for a fixed area, and it can be used to detect long time duration stops. So, we revised the algorithm based on the existing work of DB-SMoT(Direction Based - Stop and Move of Trajectory) to form the DDB-SMoT algorithm in order to better detect stop clusters in our black bear raw trajectory.

Chapter 3: DDB-SMoT, A New Spatial Temporal Clustering Algorithm

The semantic enrichment algorithm aims at annotating each stop detected in the raw trajectory with an activity to form the list of activities as the semantic trajectory. It is done by gathering the environmental information around the stop place mainly about the POIs in the region, and through the POI information we make predictions about activity performed in this area. Before we go into details about our semantic enrichment, we first need to solve the problem: how could we come up with stops with a raw trajectory. This would be the key part that will have significant impact on our accuracy of the semantic enrichment process.

The following sections will introduce a stops detection algorithm DDB-SMoT(Direction and Distance Based-Stop and Move of Trajectory) based on DB-SMoT(Direction Based-Stop and Move of Trajectory).

3.1 Basic Definition of DDB-SMoT

To better illustrate the DDB-SMoT algorithm, we should illustrate the original DB-SMoT algorithm first to see its advantages and disadvantages. The core idea for the DB-SMoT is that those consecutive stop points in the GPS trajectory always try to move around at a place which makes the direction change between direction $P_{i-1}P_i$ and direction P_iP_{i+1} tend to be larger, while the direction change among those movement points are almost small values. Base on this idea, we can select a threshold

on the direction change to identify whether the point is a stop point or a move point in the trajectory. Compared with other stops detection algorithms, DB-SMoT needs less geographic information to process and it does not require the trajectory to be inside a fixed region which makes it a very good model to develop with. One thing that DB-SMoT algorithm does require is that the trajectory must be dense enough in order to use the direction change threshold for stop clustering. If the time interval of trajectory given is relatively long, the direction change for a movement point could also be very large. As a result of that, using direction change threshold alone cannot provide us with good result of stop clustering when using the sparse trajectory as an input. In our case, to increase battery life for our GPS devices on wild animals, we can only send out signals at the average of once an hour, which makes our wild animal trajectory sparse. To improve the result, we choose to revise the DB-SMoT algorithm to let it fit to our project. In DDB-SMoT algorithm, distance change will also be considered as a threshold when filtering out move points in the trajectory which makes the algorithm gain a higher accuracy.

We will first introduce the definitions and concepts in the DDB-SMoT algorithm, and then explain the DDB-SMoT in detail.

- **Definition 1** Trajectory

  A set of GPS trajectories with their location information and timestamps: $T =$ {position:(Lon,Lat); Timestamps:(Ts);}

- **Definition 2** Direction Change

  For three consecutive points P_{i-1}, P_i and P_{i+1} in the trajectory, the direction change at P_i is the angle between the two directions $\overline{P_{i-1}P_i}$ and $\overline{P_iP_{i+1}}$, denoted

by DC(P_i).

Minimum Direction Change, denoted as $MinDirChange$, is the threshold of direction change used in the algorithm to detect Candidate Cluster Point, which will be illustrated in Definition 5.

- **Definition 3** Distance Change

  For each two consecutive points P_{i-1}, P in the trajectory, the distance change at P_i is the distance on earth between the two GPS points P_{i-1} and P_i, denoted by DisC(P_i).

  Maximum Distance Change, denoted as $MaxDisChange$, is the threshold of distance change used in the algorithm to detect Candidate Cluster Point, which will be illustrated in Definition 5.

- **Definition 3** Time Change

  For each two consecutive points P_{i-1}, P in the trajectory, the time change at P_i is the timestamps difference between the two GPS points P_{i-1} and P_i, denoted by TimeC(P_i).

  Maximum Time Change, denoted as $MaxTimeChange$, is the threshold of time change used in the algorithm to filter out those conditions that two consecutive GPS points have very large time gap when detecting Candidate Cluster Point.

- **Definition 4** Candidate Cluster Point

  In a sub-trajectory P_{i-1}, P and P_{i+1}, GPS point P_i is a Candidate Cluster Point if $DC(P_i) > MinDirChange$ and $DisC(P_i) < MaxDisChange$, otherwise, P_i should be considered as a move point in the move clusters.

- **Definition 5** Connected Candidate Point

  Let $< P_i, P_{i+1}, P_{i+2}, ..., P_{i+n+1} >$ be a subtrajectory. Then, the point P_i is a connected-candidate-point to P_{i+n+1} with respect to $MinDirChange$, $MaxDisChange$ and $MaxTol$ if P_i and P_{i+n+1} are candidate-cluster-points and $n \leqslant MaxTol$.

  The maximal tolerance threshold $MaxTol$ specifies the maximum number of trajectory points with direction change less than the $MinDirChange$ threshold that can be found consecutively in a cluster.

- **Definition 6** Trajectory Cluster

  A cluster $C =< P_i, P_{i+1}, P_{i+2}, ..., P_{i+n} >$ of a trajectory T with respect to $MinDirChange$, $MaxDisChange$, $MaxTol$ and $MinTime$ is a non-empty sub-trajectory of T formed by a set of contiguous time-space points such that:

  1) $\forall p, q \in T$: if $p \in C$ and p is a connected-candidatepoint to q with respect to $MinDirChange$ and $MaxTol$, then $q \in C$.

  2) $\forall p, q \in C$: p is connected-candidate-point to q with respect to $MinDirChange$, $MaxDisChange$ and $MaxTol$. 3) $t_n t_1 \geqslant MinTime$, where $P_i = (x_i, y_i, t_i)$ in a cluster.

3.2 DDB-SMoT Algorithm Description

Stops Detection Algorithm takes in the raw GPS trajectory with timestamps, along with the a list of pre-compiled parameters, to compute a set of stops and moves. There are two parts in the algorithm output. The first part describes the stops, which contains a list of stop clusters and each of the stop clusters contains a set of

consecutive GPS points. The second part shows the moves. It contains a list of move path and each of the movement path is consist of multiple consecutive GPS points.

The stops detection process can be divided into three steps:

1) Preprocess step: Compute the list of distance change, time change and direction change for each GPS point in the trajectory T, denoted as *variations*, *timeChanges* and *distances* in the below pseudo-code.

2) Clustering step: Find all the stop clusters in the trajectory T to set up the stops for output.

3) Postprocess step: Find all the points in the trajectory T that are not in the stop cluster to form the movement paths.

All of these steps are shown clearly in the Stop Detection Algorithm pseudo-code in Algorithm 1.

From the above explanation, the core function of the stops detection is the Find-Clusters method in the pseudo-code. Algorithm 2 is the pseudo-code and explanation for this method.

The FindCluster method starts with checking the distance change and direction change between each two points of the trajectory T. While at the point i, when the variation is larger than the direction threshold $MinDirChange$ and the distance is less than the distance threshold $MaxDisChange$, the point P_i can be added into the current stop cluster. When a point P_i in the trajectory does not variate the these two conditions, we will try to find the next point P_j with $DC(P_j) > MinDirChange$ to check whether $j-i < MaxTol$ or not. Meanwhile, we should also make sure $DisC(P_j)$ $\leqslant MaxDisChange$ and $TimeC(P_j) \leqslant MaxTimeChange$ when we look ahead to try to find the next valid candidate point. Because once a point breaks any of these two

Algorithm 1 Stop Detection Algorithm

INPUT:
$T\{P_1, P_2...P_n\}$ //Trajectory T
$MinTime$ //Minimum Time Duration for the stop cluster
$MinDirChange$ //Minimum Direction Change
$MaxTimeChange$ //Maximum Time Change
$MaxDisChange$ //Maximum Distance Change
$MaxTol$ //Maximum Tolerance
OUTPUT:
S //Set of Stops
M //Set of Moves
METHOD:
n=sizePoint(T);
//Calculate all the Direction Change, Time Change and Distance Change
for i from 2 to n **do**
variations[$i-1$]=DC(i)
distances[$i-1$]=dis(i-1,i)
timeChanges[$i-1$]=TimeC(i-1,i)
end for
//Clustering
Clusters = findClusters(T,$MinTime$,$MinDirChange$,$MaxTimeChange$,
$MaxDisChange$,$MaxTol$)
//Find Moves
for i from 1 to n **do**
if P_i is not a Stop **then**
Move = Move + P_i
else
$M = M$ + {Move}
Move = {}
end if
end for
ENDMETHOD

Algorithm 2 FindClusters

```
METHOD:
i=1;n=sizePoint(T);clusterOpened=false;
AllClusters = {}; Cluster = {}
while i <= n do
  if (variations[i] > MinDirChange AND distances[i] < MaxDisChange AND
  timeChanges[i] < MaxTimeChange) then
    clusterOpened=true
    Cluster = Cluster + P_i
  else
    if (clusterOpened) then
      // If there is a cluster
      //check direction of the next point
      nextIndex=lookAhead(MaxTol,MinDirChange,MaxTimeChange,MaxDisChange)
      if (nextIndex < MaxTol + i) then
        // add the points to the cluster
        for j from lastIndex to i do
          Cluster = Cluster +P_j
        end for
        i = nextIndex
      else
        //Close the cluster
        if (time(Cluster)> MinTime) then
          //Record the cluster only when the time is longer than MinTime
          AllClusters = AllClusters + {Cluster}
        end if
        Cluster = {}
        clusterOpened = false
      end if
    end if
  end if
  i++
end while
return AllClusters
ENDMETHOD
```

conditions while we look ahead, we should close the current stop cluster at the original point before we look ahead. If we have a point which $TimeC(P_j) > MaxTimeChange$ when we look ahead, the two consecutive GPS points lost their inner connection because we do not have any information about where the animal goes during the time from P_{j-1} to P_j. If we have a point on which $DisC(P_j) > MaxDisChange$ when look ahead, the two consecutive GPS points apart from each other too far to be into the same stop cluster. After the look ahead process, if we are able to get a point and it is within the $MaxTol$, which means the points between between P_i, P_j can be tolerated, we can then add all them to the current stop cluster. Otherwise, the current point P_i will be considered as a move point and thus close the current stop cluster. When closing the current stop cluster, the algorithm will check whether this stop cluster can be considered as a valid one by checking the time duration for the current stop with the time threshold $MinTime$.

The time complexity of the algorithm is O(P), where P the number of trajectory points, since each point of trajectory is analyzed only once.

Example: Below is the example to describe how this algorithm works. Suppose we have a GPS trajectory $< P_1, P_2, ..., P_9 >$ with all the parameters listed in the figure and time interval between each two points are 1 hour, we will use it to calculate the stops in the trajectory. To simplify the situation, We assume the time change between points are all valid. To start with, the algorithm calculate all the direction changes and distance changes, which has already been annotated on the path. Then, through looping from P_1 to P_9, we can find out that:

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$: $DirChange > MinDirChange$ and $DisChange < MaxDisChange$, P_1, P_2, P_3 are Candidate Cluster Points

$P_4 \rightarrow P_5$: $DirChange < MinDirChange$ and $DisChange < MaxDisChange$, P_4 is not a Candidate Cluster Point, so look ahead to find the next valid Candidate Cluster Point

$P_5 \rightarrow P_6$: $DirChange > MinDirChange$ and $DisChange < MaxDisChange$, P_5 is a Candidate Cluster Point, since the $MaxTol$ is 1, we add the P_4, P_5 into the current stop cluster

$P_6 \rightarrow P_7$: $DirChange > MinDirChange$ and $DisChange > MaxDisChange$, P_6 is a not Candidate Cluster Point

$P_7 \rightarrow P_8$: $DirChange < MinDirChange$ and $DisChange < MaxDisChange$, P_7 is a not Candidate Cluster Point, we need to discard P_6, P_7 and close current stop cluster with respect to the $MaxTol$

$P_8 \rightarrow P_9$: $DirChange < MinDirChange$ and $DisChange < MaxDisChange$, P_8 is a not Candidate Cluster Point

After looping through all the points in the trajectory, we detect one stop cluster $< P_1, P_2, ..., P_5 >$ from the trajectory $< P_1, P_2, ..., P_9 >$.

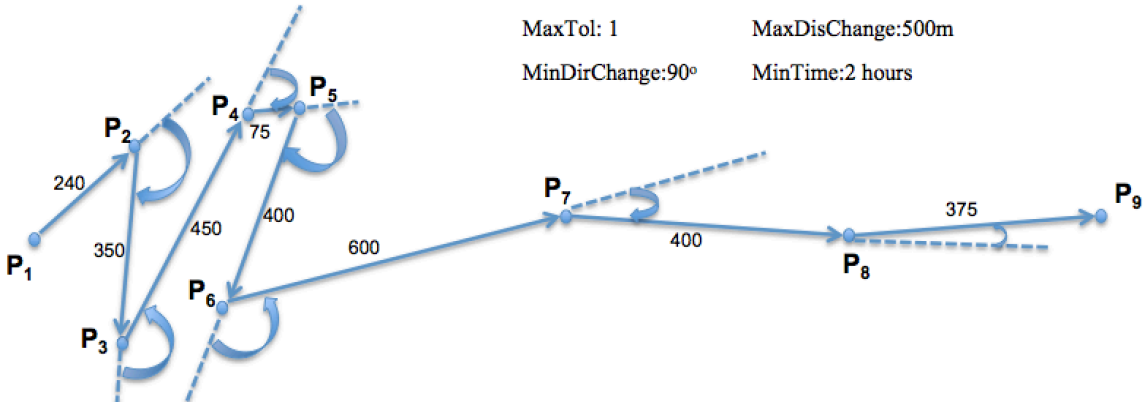


Figure 3.1: DDB-SMoT Algorithm Example

Chapter 4: Semantic Enrichment Algorithm, an Activity Identification Algorithm for Animal Trajectory

To infer the activities of wild animals, the main resource we use is the raw trajectory, which is a sequence of spatio-temporal points. The basic assumption of our approach is that the animal moves in order to get closer to the place that is interesting to it, and after it arrives the place, it will stay around to perform some activity and then move towards the next interesting place. Under this assumption, we split the raw trajectory into two parts, one part would be the movements in the trajectory, the other would be the stops.

A stop in a trajectory is identified by the absence of move and this can be detected in several ways. The segment of a trajectory between stops is called move and indicates the actual movement. A trip represents the move between two consecutive stops. With stops and moves in the raw trajectory, we can now generate a more information for each discrete points in raw trajectory. With stops and moves information, our goal is to annotate each stop with an activity. After annotating a stop with an activity, we will then gain a semantic trajectory, which consists of a list of activities performed by the wildlife on its trajectory. The process of annotating a raw trajectory with semantic information creating a semantic trajectory is called Semantic Enrichment. Our contribution is focused on the inference of the activity performed during the stops, which is the goal of its movements. In another words, we are trying to explain the reason why the animal goes and stays there.

A stop in an animal trajectory is usually associated with a place where it goes to perform some activity. In our paper, we describe such place as Point Of Interest or POI. Each POI has a name to describe it, a geographical position, including its longitude and latitude, with one or more categories to classify it and some constraints to specify its properties. An example of POI is the black bear den: the center of the den is the representative of this POI, the category can be "resting place", and the name is "black bear den".Special case of POIs are the places that are only attractive to some bears, but not to all the bears. We ignore this case and only take into account the places of interest to an all the bears or all the animals within the same species. We assume that during the stop at a POI an animal may perform an activity such as eating, drinking, resting, etc. However, at a POI an animal actually have the chance to perform different kind of activities, for example, a bear stops at a river can be considered as a "drinking" activity or as "fishing". To make this association clear, we defined a list of activities A interesting for a given species of animal, a list of POI categories C extracted from the POIs present in the tracking area, and then we mapped each POI category to an activity, thus defining a POI-Activity mapping μ. For example, consider the xxRiver(assume it is a river in MO) POI belongs to a category River. If the list of activities contains drinking we can define a mapping $\mu(River) = Drinking$, and then associating each river to a drinking activity.

4.1 Basic Definition of Semantic Enrichment Algorithm

The semantic enrichment algorithm aims at annotating each stop detected in the raw trajectory with an activity to form a list of activities as the semantic trajectory. It is done by gathering the environmental information around the stop place mainly about the POIs in the region, and through the POI information we make predictions about activity performed in this area.

The whole semantic enrichment process has two phases, the first phase is a pre-processed phase to find out the stop clusters inside the raw trajectory and gather POIs around each stop, the second phase is the process phase to come up with the most probable POI around each stop and then predict the semantic information for it.

The inputs for the semantic enrichment algorithm are:

- A POI list with their categories and other information: $POI = \{$position:(Lon,Lat); category:(C);other information$\}$
- A set of GPS trajectory with their location information and timestamps: $T = \{$position:(Lon,Lat); Timestamps:(Ts);$\}$
- A set of characteristics of the animal: Maximum Reachable Distance:(Mrd)
- A set of Activities A
- The mapping of the POI categories C to Activities A. The type and number of the activities is strongly dependent on the domain and the type of enrichment we are interested in.

- A set of spatio-temporal Domain Rules:

  Spatial Rule - Filter out all the POIs outside the range outlined by the Maximum Reachable Distance.

  Temporal Rule - Check the temporal compatibility of the arrival and departure to the stop with the valid of the POIs.

- A probability model that associates to each POI, a probability of being visited:

  P(POI_i, $stop_j$) = f(dis(POI_i, $stop_j$))

For each stop detected from the raw trajectory, we will use the spatio-temporal domain rules to filter out invalid POIs. With the spatial rules, we filter out POIs that have distances to the stop representative longer than the Mrd(Maximum Reachable Distance). Meanwhile, with the temporal rule, we filter out POIs that are not valid during the time the animal stays around the stop center. That is when the animal stays around at this stop, it is not the right time for it to be at this POI. After filtering out those invalid POIs for each stop, we use the probability model to come out with the most probable POI to link with the stop. Then, with the category to activity mapping, we can return the most probable activity for the stop.

As showed in the Figure 4.1, the procedure of the semantic enrichment process after we get raw GPS trajectories would be in three steps:

Step1: Use stop detection algorithms to identify stops in the spatial-temporal points

Step2: With the output from Step 1 and set of POIs, extract the set of POIs that are probable for bears to stay

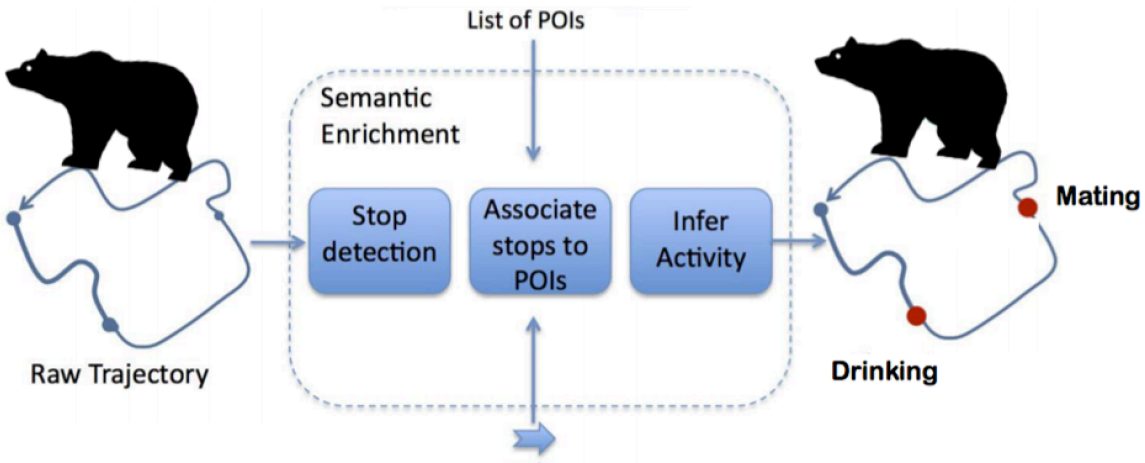


Figure 4.1: Semantic Enrichment Process

Step3: Use the set of POIs and Category with a probability model to select the most probable activity

The semantic enrichment process has been implemented into the SemanticEnrichment algorithm, described with pseudocode in the next section.

4.2 Semantic Enrichment Algorithm Description

The semantic enrichment process can be divided into steps as shown in Figure 4.1. The main procedure of the semantic enrichment algorithm is presented by the pseudocode below. The algorithm takes in the GPS trajectory to output the semantic trajectory as a list of activities performed in each step of the raw trajectory.

The first step of the algorithm is to detect the stops in the raw trajectory. With the DDB-SMoT algorithm we explained in the chapter 3, we can get a list of stops from the raw trajectory. Secondly, we use the stops to link with POIs and then predict the activity. For each stop, with the spatio-temporal rules, we can filter out invalid

POIs from the POI list. Finally, with the remaining POIs for each stop, we use the probability model to compute the most probable POI and map it to the corresponding activity.

Algorithm 3 Semantic Enrichment

```
INPUT:
 T{P_1, P_2...P_n}     //Trajectory T
OUTPUT:
 ActivityList          //Set of Activities
METHOD:
//Detect Stops from the raw trajectory
Stops = StopDetection(T);
for all (s in Stops) do
   //Select POIs for the stop s
   possiblePOIs = selectPOIs(s,MaxDistance);
   //Select the most probable activity and add it to list
   Activity = probability(possiblePOIs);
   ActivityList = ActivityList + {Activity};
end for
return ActivityList
ENDMETHOD
```

Once all the stops can be detected in the trajectory, the enrichment process will then link each stop with a set of potential POIs, and choose the most probable POI from the set to infer the activity performed during this stop.

The pseudo-code in Algorithm 3 shows how to select the POIs from the POI list for each stop. It takes in the stop information and output a set of POIs associate with the stop. With more information about the POIs, we definitely can filter out more POIs to get a more appropriate list of POIs, unfortunately, we are not able to get information for wildlife animal POIs that is suitable for filtering. As a result of that, we will simply use the distance as a threshold to filter out some POIs. We defined the $MaxDistance$ to filter out those POIs that has the distance to the stop

representative further than the $MaxDistance$ threshold. The $MaxDistance$ should be computed with respect to the speed when the animal is in a stop and the average time interval between each two consecutive GPS points in the trajectory.

Formally, A POI p for a stop s is selected if $d(p, s) < MaxDistance$, where d is a function that returns the walking distance between two locations on earth, $MaxDistance$ is a parameter that depends on the speed in a stop and time interval between points.

Algorithm 4 SelectPOIs

```
INPUT:
 S                //Stop S
OUTPUT:
 selectPOIs       //Set of POIs
METHOD:
selectPOIs = {};
for all (poi in POIList) do
  //Select POIs for the stop s
  if (distance(S,poi) < MaxDistance) then
    selectPOIs.append(poi);
  end if
end for
return selectPOIs
ENDMETHOD
```

SelectPOIs algorithm above shows the detailed procedure of retrieving all the selected POIs for a given stop.

The probability computation step measures for each selected POI, the corresponding probability of being visited starting from the stop. We consider a method based on the Gravity model below.

The probability model to infer the most probable category is based on the Newton's Law of Gravitation and used to predict the degree of interaction between two

places. This degree is proportional to the masses and inversely proportional to the square distance between them, the well known formula for the Law of Gravitation is represented by :

$$GravLaw = \frac{mass1 * mass2}{distance^2} \tag{4.1}$$

This is exactly what we need to evaluate the attractiveness of a POI to a stop. In another word, the probability model should emphasizes both the distance from the POI to the stop and also the masses of the POI to the stop.

Since the goal for our semantic enrichment process is to annotate an activity with each stop, we do not need to find out which exact POI the wild animal is at when it stops. For the list of the potential POIs that linked to a stop, the process will group them by POI category and evaluate the probability for each POI category. In this case, instead of giving each POI a probability, the process will assign each POI category with a probability, so that within a POI category, each potential POI will share the same probability. And thus, with the help of the category activity mapping, we can produce the most probable POI category to map to the activity.

We instantiate the original definition of the Gravity model using the principle of bodies attraction where $mass1$ represents the point of stop - to which we give value 1 by definition, and $mass2$ represents the mass of the POI categories. In another word, $mass2$ represents the number of POIs in the reachable POIs that can be mapped to the given POI category. Here is the revised formula for the probability model:

$$P(s, act) = \frac{|p \in SelectedPOIs(s)|(p.category) = act|}{min(d(s, p))^2} \tag{4.2}$$

Where the function SelectedPOIs returns the POIs selected using the Selected-POIs algorithm given the stop s, p.category indicates the category of POI p and d(s,p) is a function returning the distance between the stop and the set of POIs p associated to the same activity. This equation uses the minimum distance of all the distances calculated from the stop to each POI position in the selected POIs as the representative distance. The selected POIs are the input for the Probability algorithm. Thus, with this model we associate a probability to each possible activity relative to the stops. With this probability model, we take into account not only the distance of the POIs from the stops, but also the characteristics of the location where the user stopped.

Algorithm 5 Probability

INPUT:
S //Stop S
POIs //Set of selected POIs
OUTPUT:
Activity //The predicted Activity
METHOD:
probability = {};
for all (*act* in *ActivityList*) **do**
//For each group of POIs mapped to the same activity
$POIs_{act} = \{p \in POIs : (p) = act\}$;
//Takes as distance the minimum among the stop and
//all the POIs mapped to the same activity
$dist$ = min(distance(S,p) for $p \in POIs_{act}$);
//Compute the mass of these POIs as the number
//of POIs of the same category
$mass$ = length($POIs_{act}$);
//Compute the gravity value for this category
//activity and add to the probability list
probability.append(act, $mass/dist^2$);
end for
return *Activity* = max(*probability*.act);
ENDMETHOD

Example: The example in Figure 4.2 below shows us the process about how to choose the most probable category to infer the activity performed at the stop. Let us suppose to have the stop and the selected POIs shown in the figure. The POIs, located at different distances from the stop, belong to categories $Category_1$, $Category_2$ and $Category_3$. These categories are mapped to activities Act_1, Act_2 and Act_3 respectively. According to the Gravity Model defined above, the probabilities for the three activities are the following:

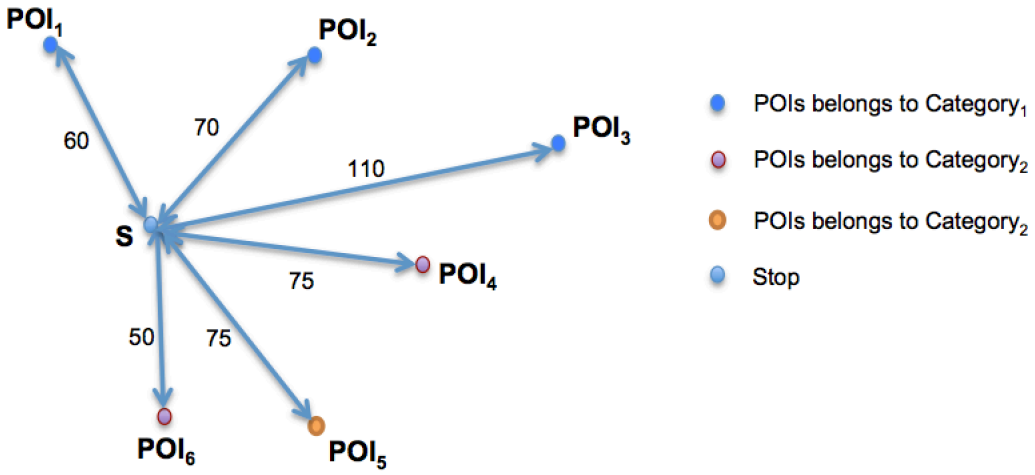


Figure 4.2: Semantic Enrichment Activity Inference Example

$P(S, Act_1) = \frac{1}{\phi} \times \frac{1}{60^2} = 0.36$

$P(S, Act_2) = \frac{1}{\phi} \times \frac{1}{75^2} = 0.52$

$P(S, Act_3) = \frac{1}{\phi} \times \frac{1}{50^2} = 0.12$

where the $\frac{1}{\phi}$ acts as the normalizer factor. From the probability results above, we can see the Act_2 has the highest probability, which means the wildlife has the highest probability to performs Act_2 at this stop.

Chapter 5: Animal Trajectory Generator Based on Stop and Move Model

In order to test the stops detection algorithm DDB-SMoT, we need a trajectory dataset with ground truth labeled at each point of trajectory to verify prediction produced by DDB-SMoT. However, such trajectory dataset does not exist for wild animals. As a result, we built a wild animal trajectory generator based on the parameter from the real wildlife GPS trajectories. The generator outputs wild animal trajectories based on the parameters it is fed and each point in the track can obtain a stop or move label according to the way it was generated. With the help of this generator software, we are able to obtain the accuracy of stop detection algorithms and evaluate their performance with different bench marks.

In the following sections, we will provide detailed explanation of how we implemented the wildlife trajectory generator, and how we use it to evaluate the performance of algorithms.

5.1 Basic Definition of Animal Trajectory Generator

To test the stops detection algorithm, we made up a wildlife trajectory generator based on the features extracted from the real black bear GPS trajectory. The generator outputs trajectory point with a stop or move label as a ground truth for the experiment. Then, we use our stops detection algorithm to run on the same trajectory

to label each point again. Finally, we compare the result from our algorithm with the ground truth that we generated with the accuracy to evaluate the whole process.

The idea of implementing the trajectory generator is based on our assumption about how an animal moves and stops. In detail, we assume that there are several POIs that the animal wants to travel to a trajectory, one by one until all the predefined POIs have been traveled. Then, in this case, the whole trajectory will be simulated by two parts:

1) Move towards next POI

2) Stop at POI to perform activity

From the idea above, when simulating a wildlife trajectory, there are three problems for us to solve: first, how to come up with the transition model to let the trajectory be able to transit from movement points to stop point and vise versa; second, how to simulate stop points to let it move around a POI center; finally, how to simulate movement points to let it move toward a POI center. Next, we will illustrate in details about some features and definitions we will use and how we solve all these problems.

- **POI:** A location of the POI along with the radius to specify the range of the POI that the wildlife is available to perform activity POI = {position:(Lon,Lat); radius:(r);}

- **Plane:** A 2D space that will be used for simulating the GPS trajectory $Plane$ = {width:(wid); length:(len);}

- **Time Interval:** A fixed time interval between two consecutive timestamps in the GPS trajectory. Since the time interval between each two GPS points from a

typical GPS device on the black bear is around a half hour, we will use half hour as the time interval when simulating trajectory.

- **Average Direction Change:** The Average Direction Change(ADC) calculated from the real black bear GPS data.
- **Direction Change Standard Deviation:** The Direction Change Standard Deviation(DCSD) calculated from the real black bear GPS data.
- **Average Speed:** The Average Speed(V) calculated from the real black bear GPS data.
- **Speed Standard Deviation:** The Speed Standard Deviation(SSD) calculated from the real black bear GPS data.
- **Stop to Stop Probability:** the probability of the next point is a stop point if the current point is a stop point. (This probability helps the generator to jump out of the stop cluster)

The average speed, average direction change, speed standard deviation and direction change standard deviation should be different for the stops model and movement model to generate different kind of tracks.

To better simulate the trajectory, we extract some features from the real black bear GPS dataset(2011-2015). First, we extract the Stop to Stop Probability from the black bear trajectory:

$$P(p.next = s|p = s) = \frac{|\{p \in T|p.next = s, p = s\}|}{|\{p \in T|p.next = s, p = s|\}| + |\{p \in T|p.next = m, p = s\}|} \tag{5.1}$$

Also, with the raw trajectory, since there is no actual ground truth provided for the GPS trajectory to label a point is a stop or move, we will use our stop detection algorithm to get the stops and moves to set up a estimated ground truth. Then, we calculate the mean speed, mean direction change, direction change standard deviation and speed standard deviation for stops and moves separately. In this way, we have two sets of four parameter. Specifically,

Parameters for Movement Model are: $\overline{V_{move}}, \sigma_{V_{move}}, \overline{DirChange_{move}}, \sigma_{DirChange_{move}}$,

Parameters for Stops Model are: $\overline{V_{stop}}, \sigma_{V_{stop}}, \overline{DirChange_{stop}}, \sigma_{DirChange_{stop}}$,

5.2 Trajectory Generator Model

5.2.1 State Transit Model

Like we explained above, there are two states in the trajectory: stop, move. In this section we will define how the generator transit from stop to move or the other way around. After we set up the amount of POIs that the animal intends to visit, we will start to simulate move points and stop points with fixed time interval.

First of all, the generator will start to use the movement model to generate points to approach the first POI. The generator continuously check the distance between the current point and the POI center position. If $dis(p, POI) > POI.radius$, the generator will try to keep move closer to the target POI center. On the other hand, when the distance between current point and the POI center is less than the POI radius($dis(p, POI) < POI.radius$), it will automatically transit into stops model to stay around and inside the POI range.

The base idea for the stops transit model is based on the Markov probability model. Each time the generator tries to simulate the next point in the stop, we first check whether the next point should be a stop. If $RandomZeroToOne() < StopToStopProbability$, we will start generate the next point as a stop with stop model. Otherwise, the generator will jump out the current stop and start to use movement model to move towards the next POI. To ensure the generator can actually simulate some points at around each POI, we first generate 5 stop points around this POI with stop model, and start to use Markov probability model to decide whether to jump to movement model from then on. With this addition condition, the generator can simulate at least 5 points in each POI for the worst case.

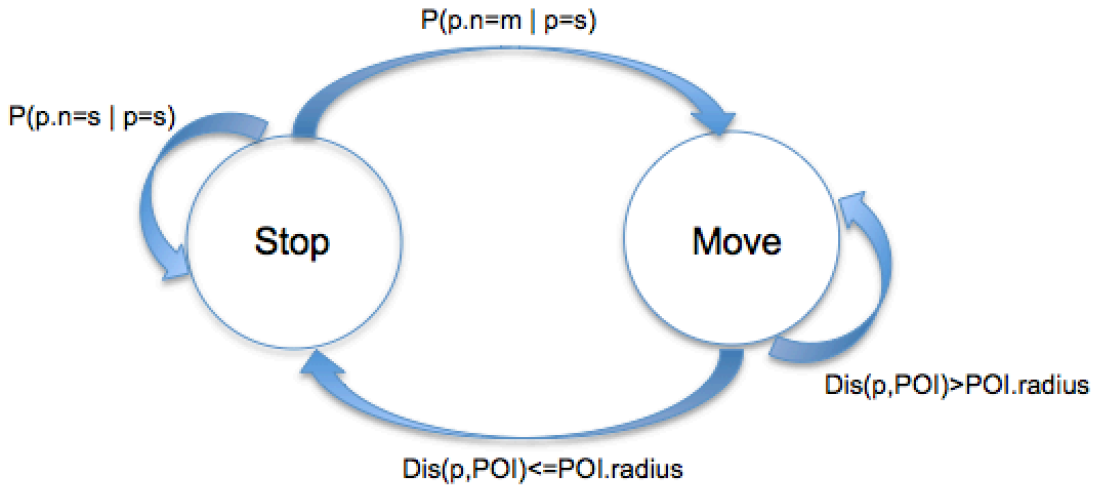


Figure 5.1: Trajectory Simulator Transition Model

The generator keeps generating points until all the POIs in the list have been visited once, or it reaches the maximum number of points threshold, whichever comes first.

The Figure 5.1 shows the state transit model for the wildlife trajectory generator. It specifies how the stop model jumps to the movement model and how movement model transits back to the stop model:

5.2.2 Movement Model

The movement model aims at keeping moving closer to the POI center until $dis(p, POI) < POI.radius$. To generate the next point in a movement model, we first set up the distance from current point to the next point, and then come up with the direction change to calculate the position of the next point.

The distance can be computed by the formula:

$$dis = N(\overline{V_{move}}, \sigma_{V_{move}}) * t \tag{5.2}$$

where t is the Time Interval of an hour(to better fit our real situation), and the speed of the wildlife from the current point to the next point will be calculated by using the Gaussian Random with the average speed and standard deviation learned from the real black bear dataset.

The direction change can be generated by the Gaussian Random function:

$$dirchange = N(\overline{DirChange_{move}}, \sigma_{DirChange_{move}}) \tag{5.3}$$

The figure 5.2 below shows an example of generating the next movement point in the trajectory(we can randomly choose from P_{next} and P'_{next}):

With the dis and $dirchange$, using equation 5.4 and 5.5 to find the next point in the movement model ($P_{pre} = (x_p, y_p); P_{cur} = (x_c, y_c); P_{next} = (x_n, y_n); d_2 = dis(P_{pre}, P_{cur}); d_1 =$

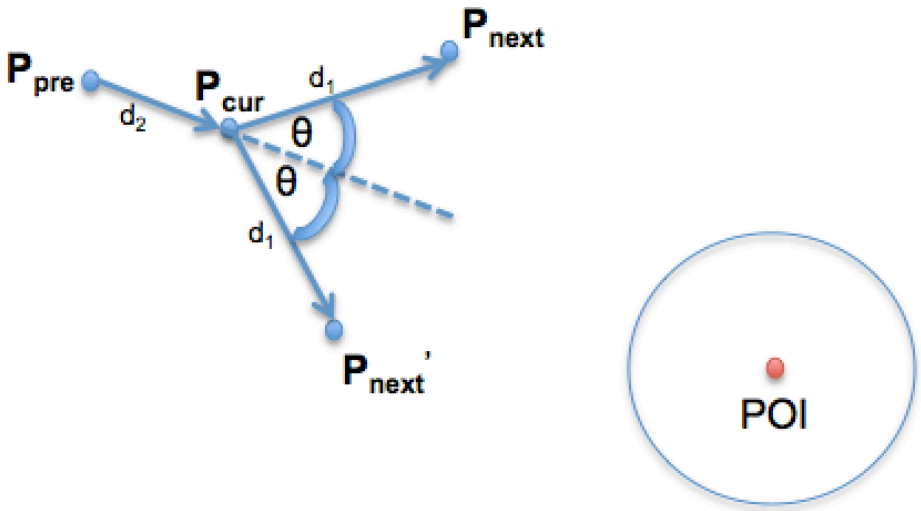


Figure 5.2: Trajectory Simulator Movement Model

$dis(P_{cur}, P_{next})$;):

$$d_2 * d_1 * \cos\theta = (x_n - x_c) * (x_c - x_p) + (y_n - y_c) * (y_c - y_p) \tag{5.4}$$

This is derived from the equation to calculate the angle between two directions in a 2D space.

$$(x_n - x_c)^2 + (y_n - y_c)^2 = d_1^2 \tag{5.5}$$

Eq (5.4) is derived from the distance equation from the current point to the next point in a 2D space.

With Eq (5.4) and Eq (5.5), we can get two pairs of (x_n, y_n) as the result. Then, we also need to compute the distance from the next point to the POI center to make sure the next point is closer to the POI than the current point. If one of these

candidate next points moves closer to the POI, we can select the one as the next. If all of the two candidate points are valid, we will randomly select one. Otherwise, we will regenerate a *dirchange* and *dis* to compute the next point again.

When the current point is very close to the POI range, it will has very low probability to get closer. So, when this condition happens, we will randomly choose a point within the POI range to work as the next point.(This only happens when the generator has tried 20 times to generate the next point but no one is a closer point to the POI than current point)

Once the distance from the current point to the center of the POI is less than the radius of the POI, the generator will jump to the stop model and start to simulate points around the POI.

5.2.3 Stops Model

The stops model aims at keeping moving around the POI center position, and maintain $dis(p, POI) < POI.radius$ until it is ready to jump out to movement model. To generate the next point in a stop model, we need to make sure that the next point is still within the range of the POI. We follow the same procedure as generating the next point in the movement model, however, we will need to add additional conditions to ensure it stays at the correct region.

To generate the next point with stop model, we still use Gaussian Distribution to generate *dis* and *dirchange*, and follow the same process in the movement model to generate the next stop point. Figure 5.3 shows how we generate it the next stop:

With Eq (5.4) and Eq (5.5), we can get two points as candidate stop points. Since the stop points need to stay within the POI range, we will check the distance from

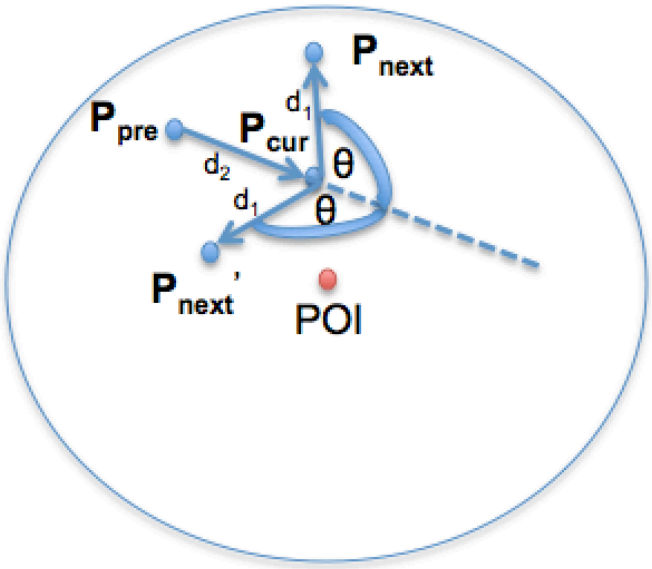


Figure 5.3: Trajectory Simulator Stop Model

the point to the POI center. If one of these candidate next points stays in the POI range, we can select the one as the next. If all of the two candidate points are valid, we will randomly select one. Otherwise, we will regenerate a *dirchange* and *dis* to compute the next point again.

The generator keeps generating stop points with our stop model until the state transition model allows it to jump out of the region and start to move towards the next POI center.

5.3 Trajectory Generator Example

The generator software is built as a Java desktop application, and currently used for generating trajectories that are similar to the black bears in Missouri.

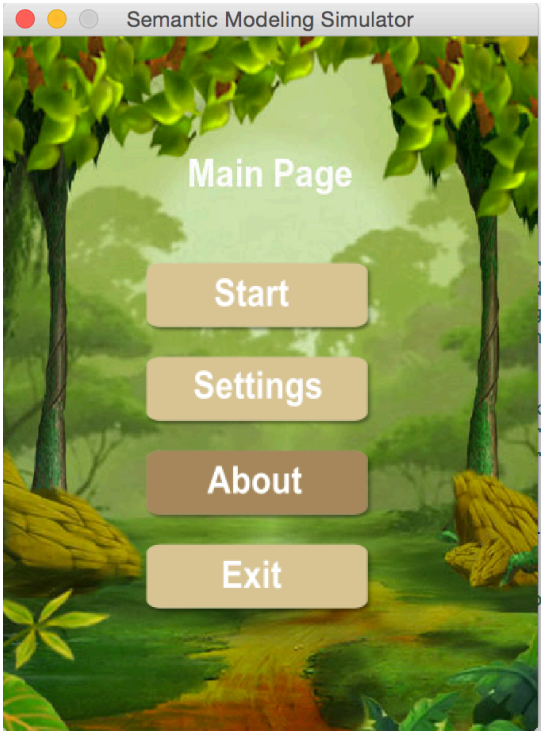


Figure 5.4: Trajectory Simulator Main Page

From the main page showed in Figure 5.4, we can start to simulate trajectory directly or set up the settings accordingly.

If it is possible to obtain parameters from other wildlife animals GPS trajectory, we are also able to simulate other animals trajectory. Figure 5.5 displays the settings table that is possible to change to generate trajectory for different purpose.

We created a software to model the trajectory, the figure shows the result of simulating trajectory. There are five POI places in the plane, and each point (red or

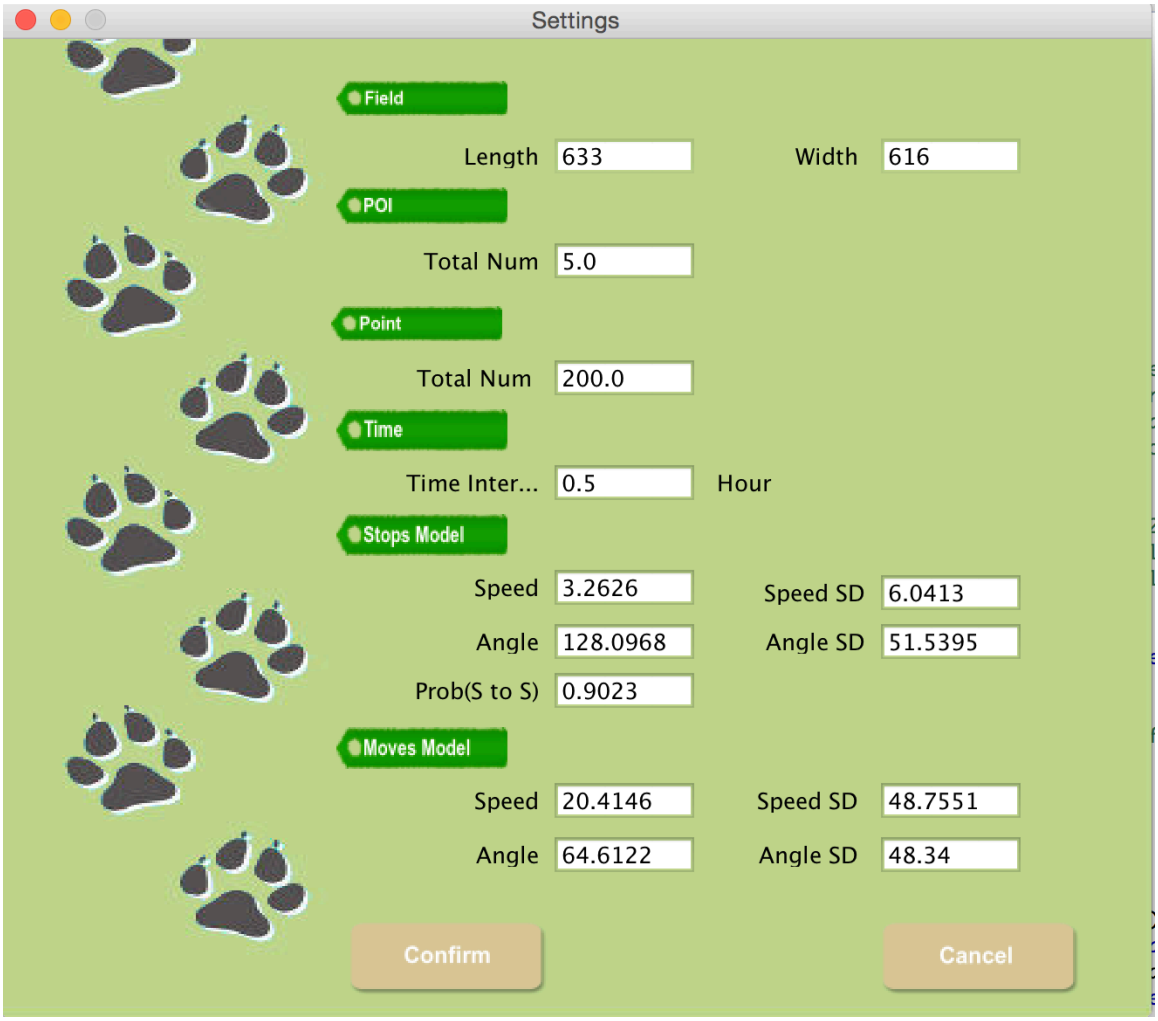


Figure 5.5: Trajectory Simulator Settings Page

green) can be treated as a GPS point with time interval of half an hour. The red points are generated by the stop model and labeled as stop point, the green points are generated by the movement model and labeled as move point. The generator provides an animation of the movement for generating the whole path. You can start, pause, stop or reset the board when simulating trajectories. The figure 5.6 below is one of the sample trajectory simulated by the software.

With the simulated trajectory, we can run our stops detection algorithm it to

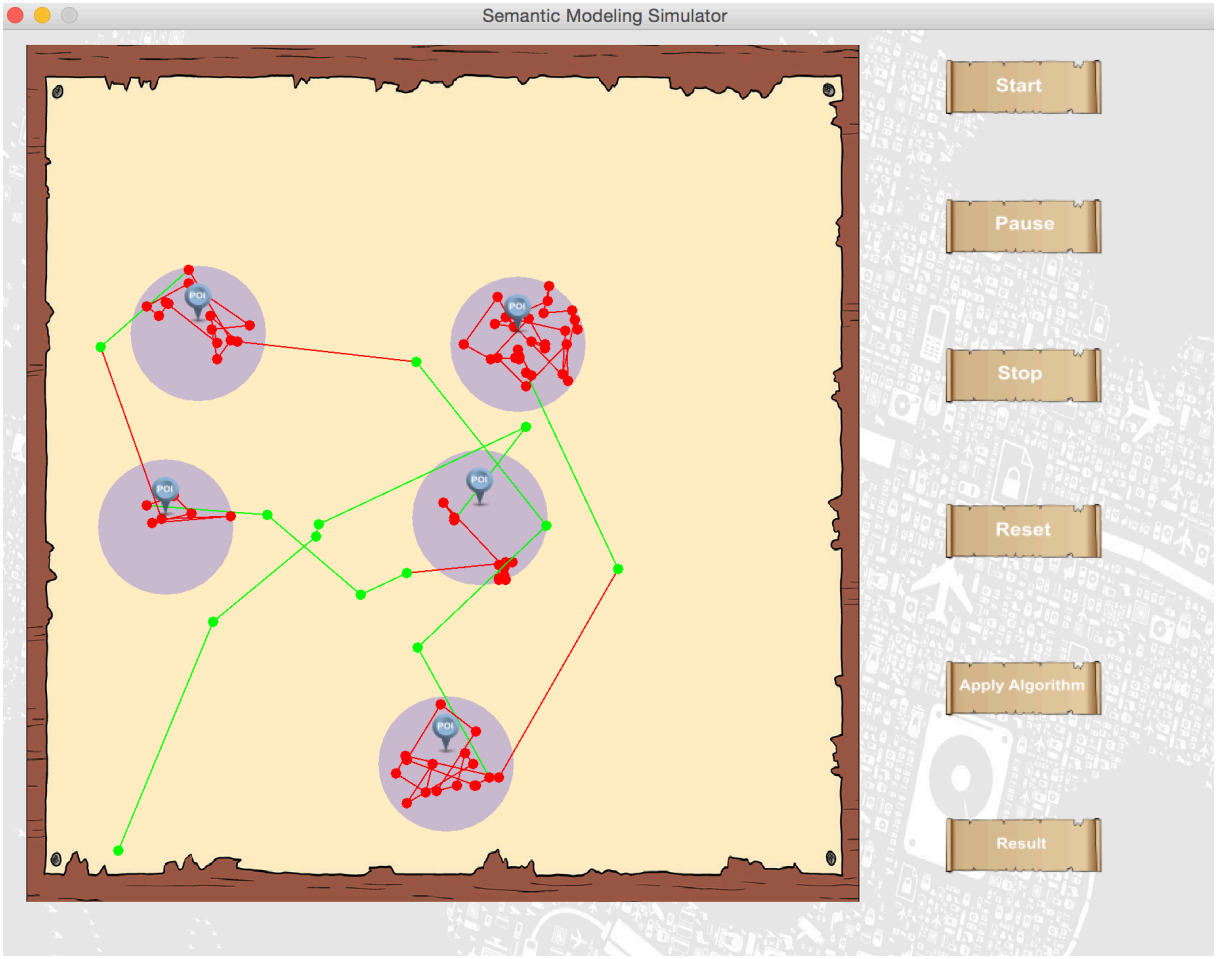


Figure 5.6: Trajectory Sample Generated by Simulator

test the accuracy of the algorithm. The figure below shows how our stops detection recognize each point in the trajectory.

The software runs the DDB-SMoT algorithm to mark each point with a "M" to present it is recognized as movement or a "S" with an index to show that it is in a stop cluster. With these labels showed on the generator board, it is clear to see whether the point is recognized correctly or not by comparing the label with the color of the point. Also, these labels for the trajectory points will be used to compare with the ground truth generated by our generator. The ground truth for each point generated

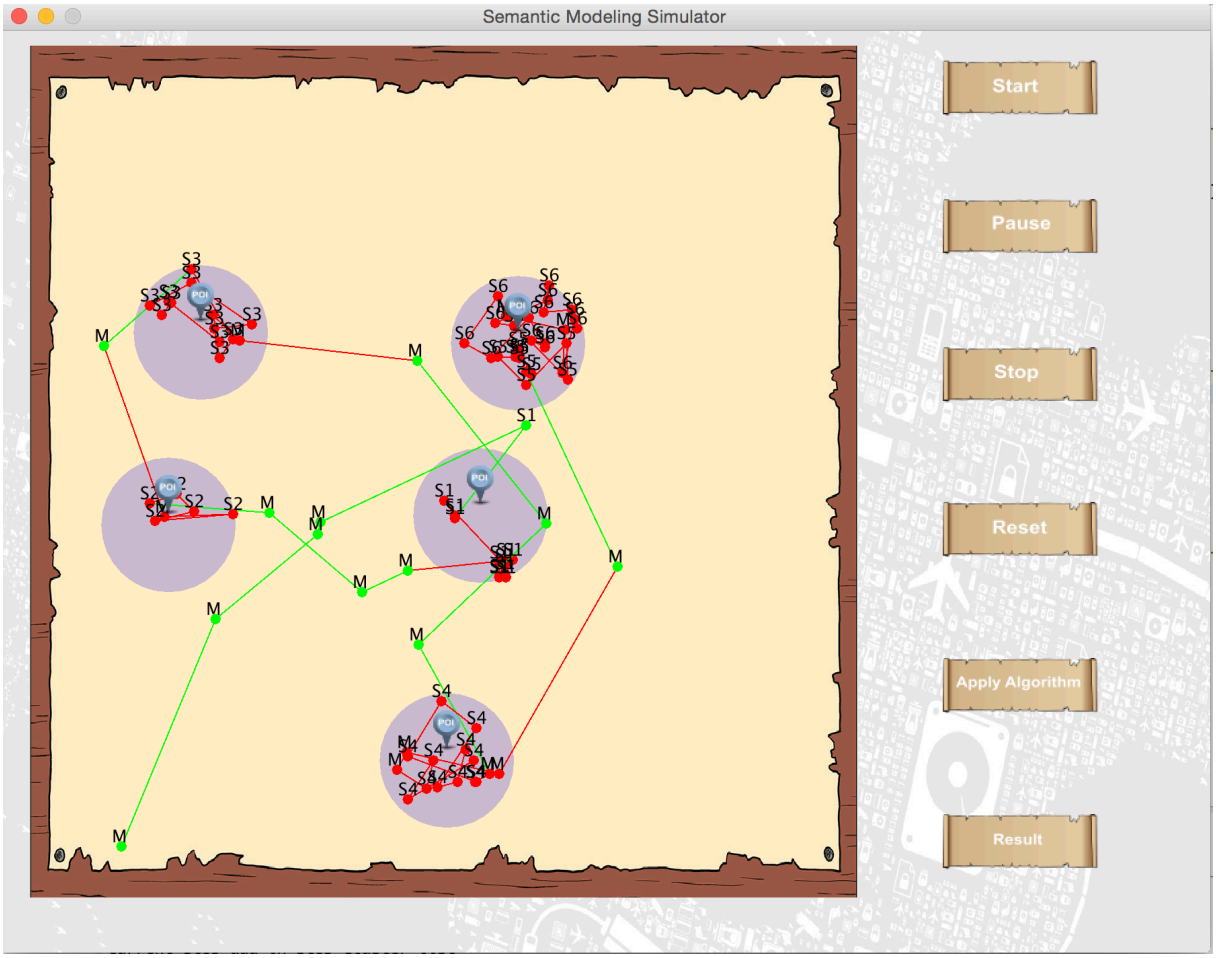


Figure 5.7: Experiment Sample Generated by Trajectory Simulator

by stop model is stop, and for the point generated with movement model is move.

After having applied the algorithm, users are welcome to click result button to show the result of this experiment. Figure 5.8 is one of the sample result extracted from our experiment:

Based on the real black bear GPS trajectory, we set up the parameters: $MinDirChange = 100, MaxDisChange = 500m, MinTime = 2hours, MaxTol = 4$

Each time the software will compare each point's predicted label with its ground truth label to check whether it is labeled correctly by our revised DB-SMoT algorithm.

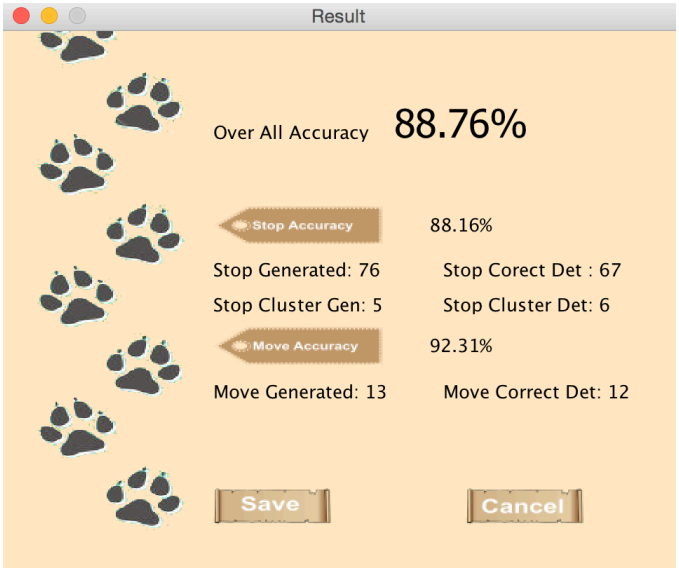


Figure 5.8: Result Generated by Trajectory Simulator

The generator can calculate:

- **Overall Accuracy** the percentage of the points that DDB-SMoT algorithm predicts correctly no matter it is a stop or move point
- **Stop Accuracy** the percentage of all the stop points generated by generator that is correctly predicted by the DDB-SMoT algorithm.
- **Move Accuracy** the percentage of all the move points generated by generator that is correctly predicted by the DDB-SMoT algorithm.

These three parameters will be used as the main bench marks when evaluating the performance of the algorithm or computing best fits for parameters in DDB-SMoT algorithm. These results will be used in our experiment in the next chapter to compare with other algorithms or evaluate the parameters that best fit our project.

Chapter 6: Experiment

There are mainly two phases in the semantic enrichment process, so we test these two phases one by one. For the stops detection phase, we built a wild animal trajectory generator based on the parameter from the real wildlife GPS trajectory. Also, we have planned to test the semantic enrichment phase on the real bear data collected from Missouri Department of Conservation, however, there are no database existed to get the POI information for wildlife. Meanwhile, it is really difficult to get the ground truth of the activity the wild animal performed. That verification of semantic enrichment will leave as future work.

In the next two sections, we will provide the experiment made for stop detection algorithm and semantic enrichment process.

6.1 DDB-SMoT Performance Evaluation

The animal trajectory generator is the main tool we used to test our revised stop detection algorithm DDB-SMoT. The features we used to form the model is extracted from the Missouri black bear GPS data provided by the MDC (Missouri Department of Conservation). the data records the GPS location of 80 bears mainly from 2011 to 2015 with more than 100000 points. With parameters learned from this real dataset applied on the trajectory generator which has been illustrated in the last chapter, we generated GPS trajectories that are similar to the real Black Bear trajectory. To evaluate the performance of DDB-SMoT, we will compare the accuracy of the result from DDB-SMoT with the result from the original DB-SMoT algorithm. In addition,

we will explain the impact for each parameter in DDB-SMoT algorithm to find the parameters that best fits our dataset.

Different wild animal trajectories has different features, we cannot use the same parameters to monitor the trajectory, nor can we use the same parameters in the stops detection algorithm to predict the stops and moves in it. With the black bear trajectory generator, we try with a different set of parameters to understand how parameter affects our prediction result and to come up with a valid set of parameters that fits with these kind of animal trajectory to come up with the accuracy of DDB-SMoT algorithm. There are three main parameters that affects the outcome of DDB-SMoT: $MaxTol$, $MinDirChange$, $MaxDisChange$

First, we analyze the $MaxTol$ parameter in our algorithm:

We tested different $MaxTol$ values while the other parameters remain the same. Typically, the $MaxTol$ should be an integer between 0 and 10. When the $MaxTol$ is too low, we have a very low level of acceptance on the points that are not a cluster candidate point, and this makes total accuracy and the stop point accuracy very low. Because even inside a stop cluster, we cannot guarantee that every point is a valid cluster candidate point. On the other hand, when the $MaxTol$ is too high, our movement point accuracy starts to drop. Since we accept too many invalid cluster candidate point, lots of movement point will be included into the stop cluster, which is probably even worse than setting the $MaxTol$ to a very low value. Because if we use this set of parameters in the real application, it will give us stops that may contain multiple stops inside a single stop cluster.

Through experiments with $MinDirChange = 100$ and $MaxDisChange = 100$, the result in Figure 6.1 suggests that we should choose a $MaxTol$ valuethat is larger

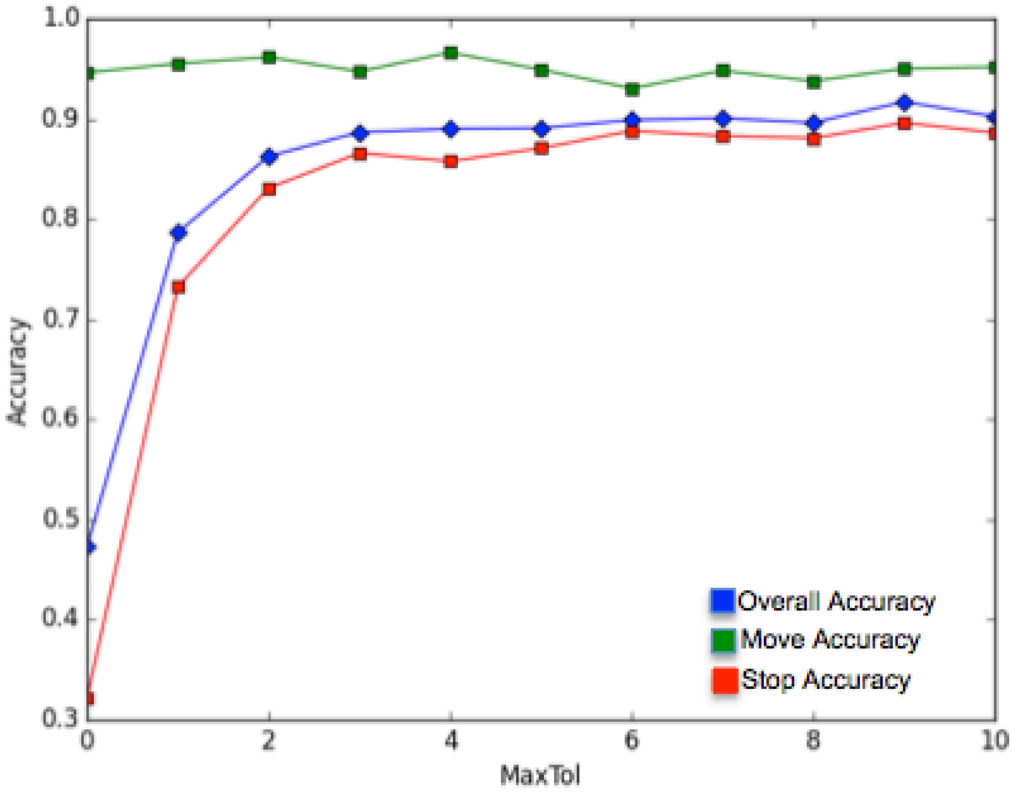


Figure 6.1: $MaxTol$ Experiment of DDB-SMoT

than 3. The result will not vary much if we choose a $MaxTol$ that is larger than 3.

Second, we try to get a suitable $MinDirChange$ for our project. To test a proper $MinDirChange$ for the stops detection algorithm, we also try with different ranges of $MinDirChange$ threshold. The figure 6.2 shows the result of using different $MinDirChange$ threshold with $MaxTol = 4$ and $MaxDisChange = 100$.

Clearly, when the direction change threshold is set too low, we will not have a high accuracy of stop cluster prediction because stop points inter mingle with movement points. Once the threshold is set too low, movement points may also become valid cluster candidate points to be added into a stop cluster which largely affects the

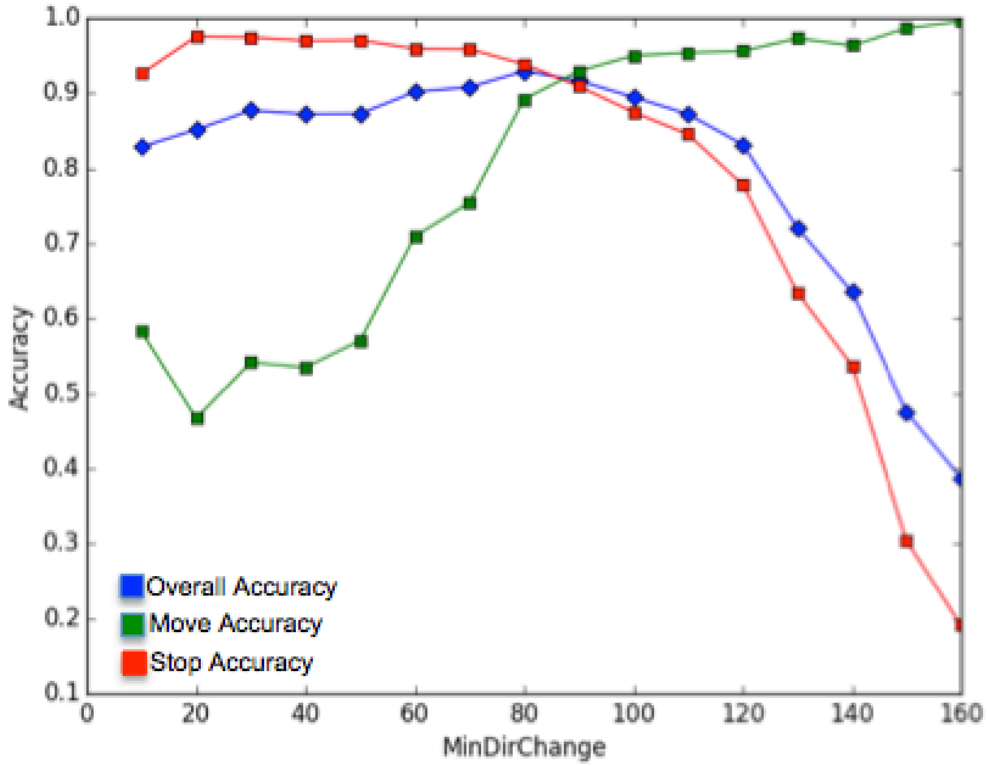


Figure 6.2: *MinDirChange* Experiment of DDB-SMoT

accuracy of the algorithm. Also, when the *MinDirChange* threshold is set too high, there is no difference between the stop points and the movement points, the algorithm will treat almost each point as a movement point. So, the movement points correct percentage is almost 100%, while the correctness of stop points is very low. From the above figure, it suggests us to choose a proper value between 80 to 120 for the correctness of stop points, move points and all the points are in a relatively high range.

Third, since the threshold for the distance is also a key factor affecting the accuracy of the algorithm, we also need to check its validity and how it affects the accuracy.

The extreme case is that we set this $MaxDisChange$ as high as possible, and our algorithm becomes the DB-SMoT algorithm since it will not filter out any points based on the distance threshold. In the real world applications, we will need to try to learn this parameter from the GPS sample data through the speed of stop clusters and speed of movement. Figure 6.3 is the result of this experiment.

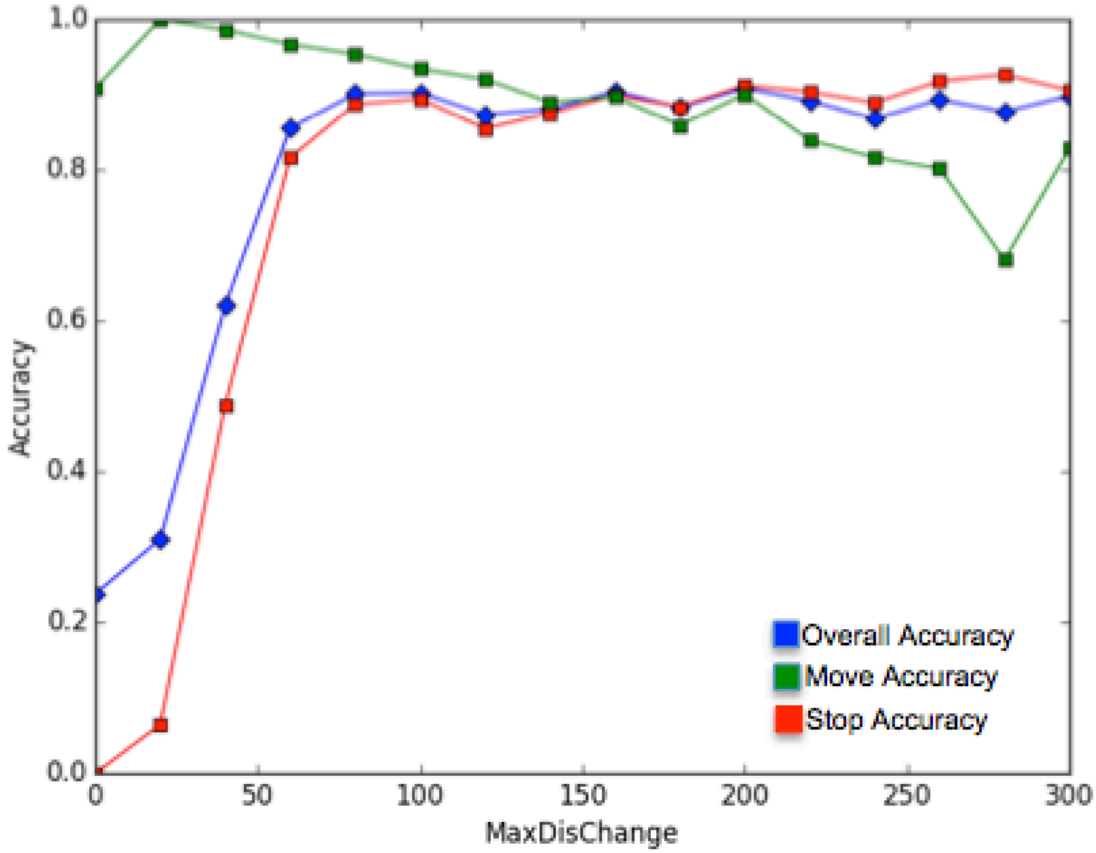


Figure 6.3: $MaxDisChange$ Experiment of DDB-SMoT

In the trajectory generator, we output a plaint that each unit corresponds to 5 meters in the real world. and we test the different $MaxDisChange$ value based on the unit with $MaxTol = 4$ and $MinDirChange = 100$. The small $MaxDis$ will

filter out too much stop points, while the large $MaxDisChange$ will lose its ability of filtering. Through continuous experimenting, we find out that 100 should be a relatively good fit for our algorithm on the black bear GPS data.

Finally, to compare the DB-SMoT algorithm with the revised algorithm, We apply the DB-SMoT algorithm on the trajectories generated by our trajectory generator, and then apply our revised DB-SMoT algorithm to compare their accuracy. Table 6.1 is the result produced by these two algorithms:

Algorithm Name	Overall Accuracy	Stop Accuracy	Move Accuracy
DB-SMoT	88.73%	91.44%	78.14%
DDB-SMoT	91.18%	90.09%	93.40%

Table 6.1: DDB-SMoT vs. DB-SMoT ($MaxTol = 4; MinDirChange = 100; MaxDisChange = 100$)

We use the same value for $MaxTol, MinDC$ in both algorithms that we have found to be a suitable value for black bear trajectory. From the table 6.1, with DB-SMoT algorithm, we can obtain much higher stop accuracy if we sacrifice the movement points' accuracy. In the real world, we actually do not want the movement points to be that low, because once a movement point is labeled as a stop, it will largely affect the stop position (the center of the stop). However, if a stop point is recognized as a movement point, it will not have much of the effects on the stop center position. Since the semantic enrichment algorithm is based on stop representative position, we do want both the move points and stop points to be detected properly in order to come out with a more accurate stop representative point. From the above explanation, we can understand that the revised DB-SMoT algorithm is more suitable to our trajectory situation.

6.2 Semantic Enrichment Experiment

We have also created a web application to show the result we got from the semantic enrichment desktop application. Though we don't have any accuracy data right now to prove the correctness of our algorithm on wild animals, it can be used as a reference to give future suggestions about where the interesting places are. Once we are able to get the geographical data with related activity, we can then provide accuracy and real predictions for black bears.

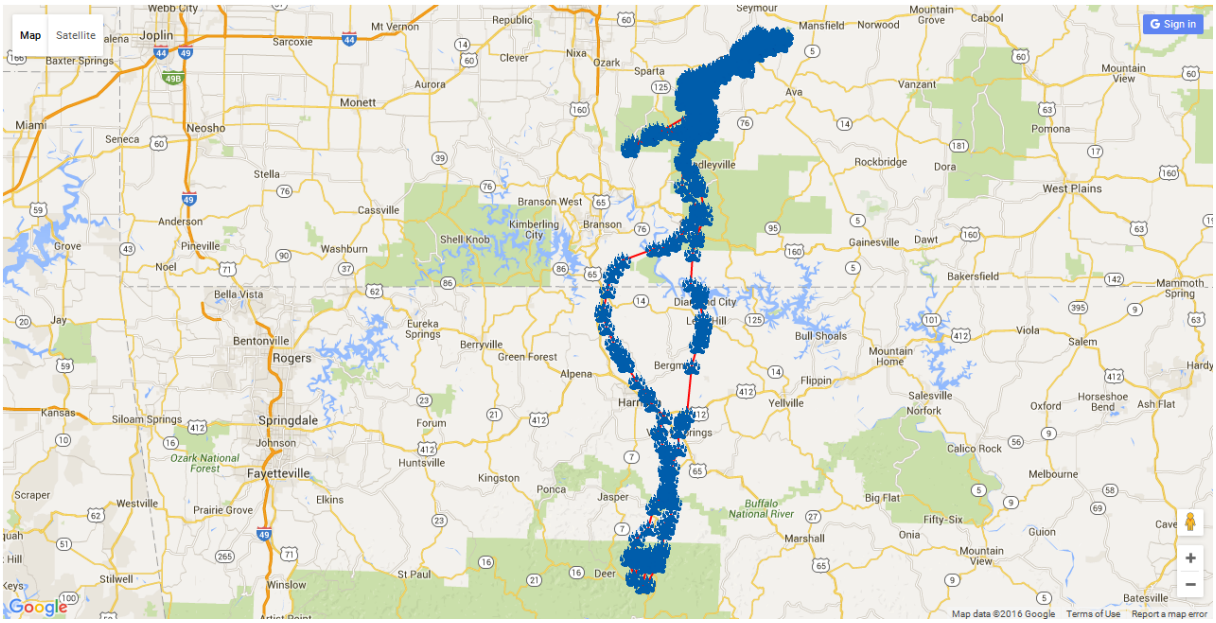


Figure 6.4: Sample Black Bear Raw Trajectory

Although we cannot directly come up with the accuracy of black bear activity performed on the trajectory, we do provide a way to understand the process of our algorithm. The web application provides a platform for users to at least visualize the process we are doing for the semantic analysis.

Figure 6.4 is the Google Map on web page showing the sample raw GPS trajectory

from real black bear GPS data. It is a single bear trajectory collected with time interval around an hour.

After the stop detection, we only keep the stop points. Compared with the raw trajectory, we get fewer GPS points there from the figure 6.5. And those points represents the key area that the bear stays, and will be used for activity identification:

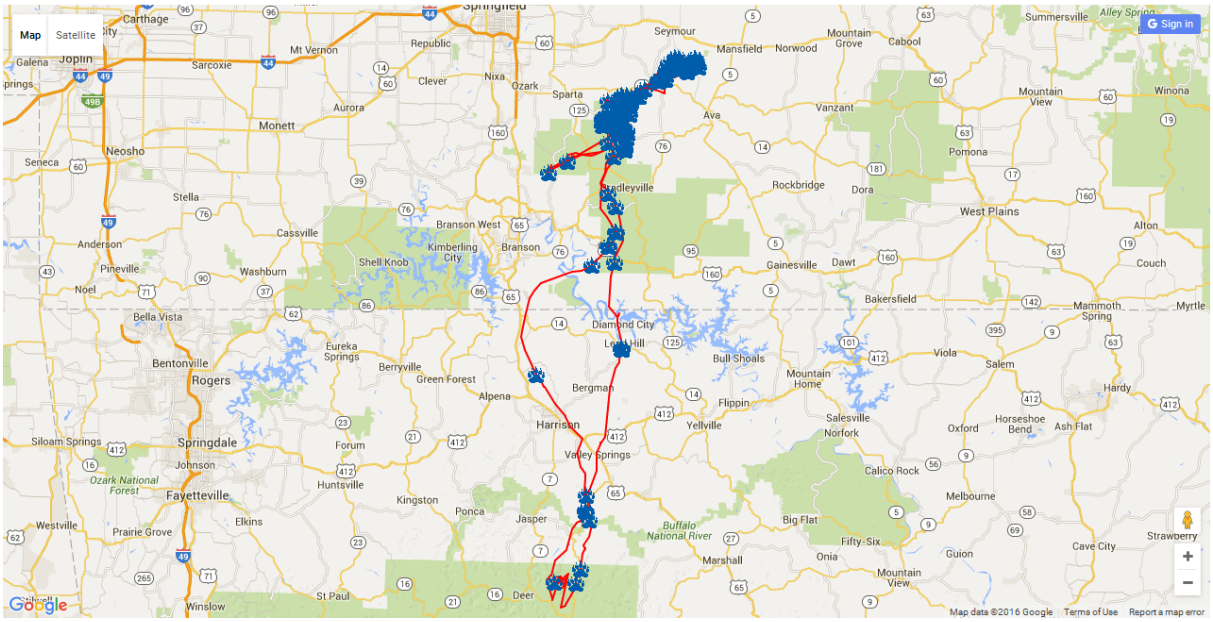
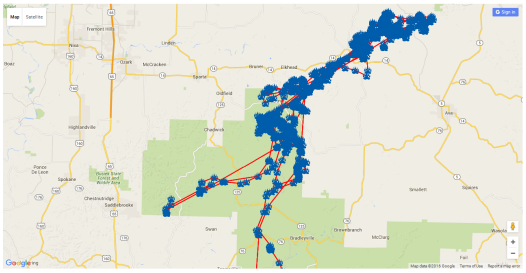


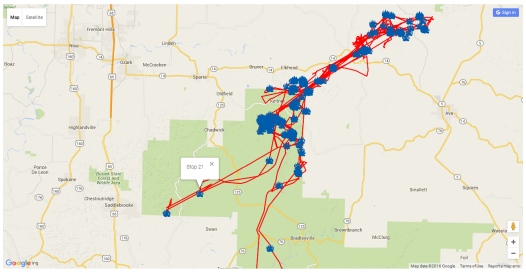
Figure 6.5: Sample Black Bear Raw Trajectory Stop Detection Result

Also, we mark each stop point with the stop cluster index just like the example in Figure 6.6(b). From the explanation about the semantic enrichment algorithm in Chapter 4, we are using a stop representative to match up with a set of potential POIs. At this point, we use the center position of all stop points that has the same stop index to represent the stop.

Since we are not able to obtain the rich POI dataset currently, we use some fabricate data with position and category information to associate each stop center with a most probable POI to infer the activity. The activity list contains Drinking,



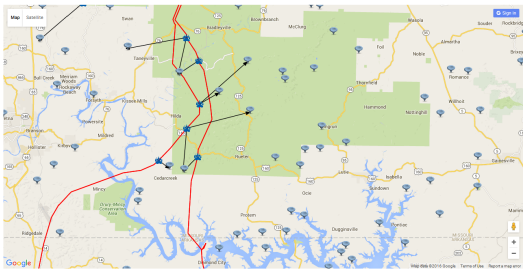
(a) Part of sample raw GPS trajectory



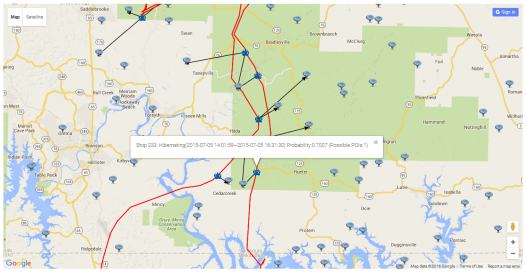
(b) Stop detection result with stop index

Figure 6.6: DDB-SMoT($MaxTol$: 4; $MinDC$: 100; $MaxDS$: 100)

Eating, Fishing, Mating, Resting, Breeding, Hibernating, Visiting(Family). Figure 6.7 is the final result of our semantic enrichment process:



(a) Semantic Enrichment Result



(b) Meanings for Semantic Enrichment Result

Figure 6.7: Revised DB-SMoT($MaxTol$: 4; $MinDC$: 100; $MaxDS$: 100)

From the result, we are able to get the information of the probability of performing activity and the POI places that are possible. The website also provides the semantic enrichment process for deer in Missouri. The deer GPS data is also provided by the MDC (Missouri Department of Conservation). Although we do not have a rich POI dataset to predict the activity performed at each stop, we can provide position information that may related to some POI places for future researchers to form the POIs for specific animals. The stop representative positions calculated for bears or deer can be treated as a reference for people to find POIs. Once the POI dataset is accessible to us, we can use it as input to perform activity inference with our software.

Chapter 7: Summary

In this project, we implemented a software tool to automatically infer the activities of wild animals during their movement when tracked by GPS devices. This problem is particularly important since we aim at semantically enriching raw GPS trajectories with more meaningful information that can be useful in various fields such as wild animal protection and wild animal population control. The approach we take is to first detect the stop points in the raw trajectory where animals stopped to perform activities and match the stops to the possible POIs they visited. With the POI category and activity mapping, we can then find the most probable POI category to infer the corresponding activity. We have also created a web application using intermediate results to visualize the outcome of each step in the semantic enrichment process. Due to the lack of actual POI data and real labeled data with ground truth, we are not able to come up with the real prediction for the wildlife animal trajectory. However, if we have access to this data in the future, we will be able to directly give predictions for animal activity performed in the trajectory using our software tool and visualize it on the website.

Also, to provide higher accuracy of stops detection results for semantic analysis, we developed DDB-SMoT algorithm based on DB-SMoT that is used in the semantic enrichment algorithm. It uses the direction change and distance change as the main threshold when selecting candidate clustering points, with the time duration threshold to filter out stops that stays too short. The algorithm is still an $O(n)$ time complexity algorithm, and is more accurate than DB-SMoT algorithm when dealing with our wild animal dataset. DDB-SMoT algorithm shows over 90% accuracy on labeling all the

stop points and movement points when test with our wildlife trajectory generator, while DB-SMoT algorithm only has less than 80% accuracy for movement points and around 90% accuracy for stop points.

To verify DDB-SMoT algorithm, it is necessary to find the wild animal trajectory dataset with each point labeled as stop or movement to gain the ground truth. However, we do not have access to this kind of dataset. Instead, we build software to generate trajectories similar to real black bear real trajectories with stop and move models to verify our algorithm. The software can be widely used for other animals trajectories given a different set of parameters extracted from other animals GPS tracks.

Currently, with DDB-SMoT algorithm and semantic enrichment software, we are ready to make real predictions once we can get a rich POI dataset for wild animals. Several remaining issues are the objectives of current and future works. First, as we already discussed, the lack of rich POI datasets is a major problem. Therefore, we are investigating the possibility of integrating more detailed POIs datasets for wildlife animals. We may also need to build this kind of dataset on our own. Secondly, we want to better define the mapping between POI categories and activities. For wild animals, there may be another kind of attractiveness except the POIs, for example, female animal may be attractive to the male animals. As a result, the social network of animals could be another aspect that we need to take into consideration. We will refine the algorithm and model to produce better result in the future.

Bibliography

[1] V. Bogorny, C. Renso, A. R. De Aquino, F. De Lucca Siqueira, and L. O. Alvares. CONSTAnT: A conceptual data model for semantic trajectories of moving objects. Transactions in GIS, 2013.

[2] P. Cintia, R. Trasarti, J. A. Macedo, L. Almada, and Fereira C. A gravity model for speed estimation over road network. In Proceedings of HUMOComp, 2013.

[3] Balazs Csanad Csaji, Arnaud Browet, Vincent A. Traag, Jean-Charles Delvenne, Etienne Huens, Paul Van Dooren, Zbigniew Smoreda, and Vincent D. Blondel. Exploring the mobility of mobile phone users. CoRR, abs/1211.6014, 2012.

[4] L. Huang, Q. Li, and Y. Yue. Activity identification from GPS trajectories using spatial temporal POIs attractiveness. In 2nd ACM SIGSPATIAL Int. Workshop on Location Based Social Networks, 2010.

[5] P. Kiefer and K. Stein. A framework for mobile intention recognition in spatially structured environments. In BMI, pages 2841, 2008.

[6] J. A. Macedo, R. Rocha, Valeria Cesario Times, Gabriel Oliveira, Luis Otavio Alvares, and Vania Bogorny. DB-SMoT: A direction-based spatio-temporal clustering method. In IEEE Conf. of Intelligent Systems, 2010.

[7] A. Tietbohl Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In SAC, 2008.

[8] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. Macedo, N. Pelekis, Y. Theodoridis, and

Yan Z. *Semantic trajectories modeling and analysis.* ACM Computing Surveys, 45(4), 2013. to appear.

[9] Mary S. Smith and Thomas A. Butcher. How far should parkers have to walk, 2008. National Parking Association Parking.

[10] S. Spaccapietra, C. Parent, M. L. Damiani, J. de Macedo, F. Porto, and C. Vangenot. A conceptual view on trajectories. Data Knowl. Eng., 65(1):126146, 2008.

[11] L. Spinsanti, F. Celli, and C. Renso. Where you go is who you are: understanding peoples activities by places visited. In In: BMI 2010 - 5th Workshop on Behaviour Monitoring and Interpretation, CEUR Workshop Proceedings, vol. 678, pages 38 52, 2010.

[12] K. Xie, K. Deng, and Xiaofang X. Zhou. From trajectories to activities: a spatio-temporal join approach. In Proceedings of the 2009 International Workshop on Location Based Social Networks, LBSN 09, pages 2532, New York, NY, USA, 2009. ACM.

[13] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and POIs. In KDD2012, 2012.

[14] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. In UbiComp, pages 312321, 2008.

[15] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw GPS data for geographic applications on the web. In WWW, pages 247256, 2008.

[16] Arbara Furletti, Paolo Cintia, and Chiara Renso. Inferring human activities from GPS tracks. Chicago, USA, 2013. ACM.

[17] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot, A conceptual view on trajectories, Data and Knowledge Engineering, vol. 65, no. 1, pp. 126146, 2008.

[18] 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 ACM-GIS. New York, NY, USA: ACM Press, 2007, pp. 162169.

[19] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, A clusteringbased approach for discovering interesting places in trajectories, in ACMSAC. New York, NY, USA: ACM Press, 2008, pp. 863868.

[20] H. G. Hazin, F. H. V. Hazin, P. E. P. F. Travassos, F. C. de Carvalho, and K. Erzini, Fishing strategy and target species of the brazilian tuna longline fishery,from 1978 to 2005, inferred from cluster analysis, Collective Volume of Scientific Papers. International Commission for the Conservation of Atlantic Tunas, vol. 56, pp. 19421951, 2006.

[21] C. Zhou, D. Frankowski, P. J. Ludford, S. Shekhar, and L. G. Terveen, Discovering personally meaningful places: An interactive clustering approach, ACM Trans. Inf. Syst., vol. 25, no. 3, 2007.

[22] D. Birant and A. Kut, St-dbscan: An algorithm for clustering spatialtemporal data, Data Knowl. Eng., vol. 60, no. 1, pp. 208221, 2007.

[23] P. Fan, A. Mohammadian, P. C. Nelson, J. Haran, and J. Dillenburg, A novel direction based clustering algorithm in vehicular ad hoc networks, in 86th Annual Transportation Research Board Meeting, 2007.

[24] J.-G. Lee, J. Han, and K.-Y. Whang, Trajectory clustering: a partitionand-group framework, in SIGMOD Conference, C. Y. Chan, B. C. Ooi, and A. Zhou, Eds. ACM, 2007, pp. 593604.

[25] P. Laube and S. Imfeld, Analyzing relative motion within groups of trackable moving point objects, in GIScience, ser. Lecture Notes in Computer Science, M. J. Egenhofer and D. M. Mark, Eds., vol. 2478. Springer, 2002, pp. 132144.

[26] P. Laube, S. Imfeld, and R. Weibel, Discovering relative motion patterns in groups of moving point objects, International Journal of Geographical Information Science, vol. 19, no. 6, pp. 639668, 2005.

[27] H. Cao, N. Mamoulis, and D. W. Cheung, Discovery of collocation episodes in spatiotemporal data, in ICDM. IEEE Computer Society, 2006, pp. 823827.

[28] J. C. Sainsbury, Commercial Fishing Methods: An Introduction to Vessel and Gear. Surrey: Fishing News (Books) Ltd, 1971.