**Design and Development of Software Agents for Location**

**Privacy-risk estimation**

Shashank Sharma

August 2018

**Abstract**

The usage of mobile devices has become ubiquitous in today’s world. The human geographical location is shared with different applications on a mobile device. The sharing of location is active not only when the application is used, but also share the user whereabouts with the third-party applications in the background. This location is used by different applications for advertising, finding new friends, suggesting new point of interests based on user trends. This location data can also be used by different third-party applications to predict user’s future locations. This is a privacy attack which makes use of user’s past location data.

In this thesis we present an algorithm which predicts user future movements with confidence percentages. This algorithm is first implemented on python using Microsoft Geolife data. This data contains 182 user trajectories data for 5 years. The same algorithm is then implemented on Android device.

The raw trajectories are used from Microsoft Geolife data. Markov chain is formed on this data to simulate a model how user data can be used to predict future locations.

**Acknowledgement**

Contents

[**1 Introduction** 5](#_Toc527911097)

[1.1 Motivation 5](#_Toc527911098)

[1.2 Design 5](#_Toc527911099)

[**2** **Related Work** 7](#_Toc527911100)

[**3** **System Model** 10](#_Toc527911101)

[3.1 Components 10](#_Toc527911102)

[3.2 Assumptions 12](#_Toc527911103)

[3.3 Problem Statement 12](#_Toc527911104)

[**4** **Markov Chain Model** 13](#_Toc527911105)

[4.1 Markov Model for Location Prediction 13](#_Toc527911106)

[**5** **Implementation** 16](#_Toc527911107)

[5.1 Variables Used 16](#_Toc527911108)

[5.2 Stay-points 16](#_Toc527911109)

[5.2.1 Algorithm 18](#_Toc527911110)

[5.2.2 Implementation Result 20](#_Toc527911111)

[5.3 States Formation 20](#_Toc527911112)

[5.3.1 Algorithm 21](#_Toc527911113)

[5.3.2 Implementation Result 22](#_Toc527911114)

[5.4 State Weights 22](#_Toc527911115)

[5.4.1 Algorithm 23](#_Toc527911116)

[5.4.2 Implementation Result 23](#_Toc527911117)

[5.5 Forming Markov chain 23](#_Toc527911118)

[5.5.1 Algorithm 24](#_Toc527911119)

[5.5.2 Implementation Result 24](#_Toc527911120)

[5.6 Android implementation 25](#_Toc527911121)

[**6 Evaluation** 27](#_Toc527911122)

[6.1 User data analysis 27](#_Toc527911123)

[6.2 Discussion and Summary 29](#_Toc527911124)

[**7 Conclusion and Future Work** 30](#_Toc527911125)

[7.1 Summary 30](#_Toc527911126)

[7.2 Future Work 30](#_Toc527911127)

# **1 Introduction**

The Location Based Applications have become very popular in last few years. There are many applications in our mobiles and computers which uses location data for its better suggestions and personalized advertisements. These applications use user location data while the application is active and inactive. Most applications have in their terms and condition mentioned about the usage of the location data.

These applications are often using user location to provide better personalized experience. The geographic location can be shared from many sources like Global System for Mobile Communication, Global Positioning System, Wi-Fi network location and so on.

## Motivation

The location-based data can tell a lot about user. It can easily answer questions like, where does user live/work? Where is user’s club/gym? What restaurant he likes? Where is he on weekends? Which hospital has he/she been visiting? All the answers to these questions can give an insight about user’s private life and whereabouts. It can clear user movement trends, his/her favorite places, his/her lifestyle and so on. The user movement patterns are found using the user’s location data for few weeks. With continuous learning, this data can be updated and have the actual user home location and work location updated with time.

This private data is often distributed to third parties. This could be an attack on user’s private life, compromising user privacy and sharing his data with other applications, friends and family. With the advancement of Artificial Intelligence and Machine Learning, it has become ever easier to exploit this data, understand bulk geographic data and infer meaning from several locations.

## Design

In this thesis, we present a location prediction model with confidence level. The location data used is from Microsoft Geolife data of 182 users for 5 years. This data is used to form Markov chain model and hence used to predict user movements based on known locations.

This algorithm is designed to predict human locations in a real-world scenario. The Geolife data is taken as input and then processed using the below algorithm.

The Algorithm has several steps:

* Detect stay-points (also detect start or end of the trajectory)
* Extend leaving time from one location and arriving time to another location based on speed and distance between them
* Group stay-points to form states
* Create time-slotted data from the states
* Apply Markov chain for the data available

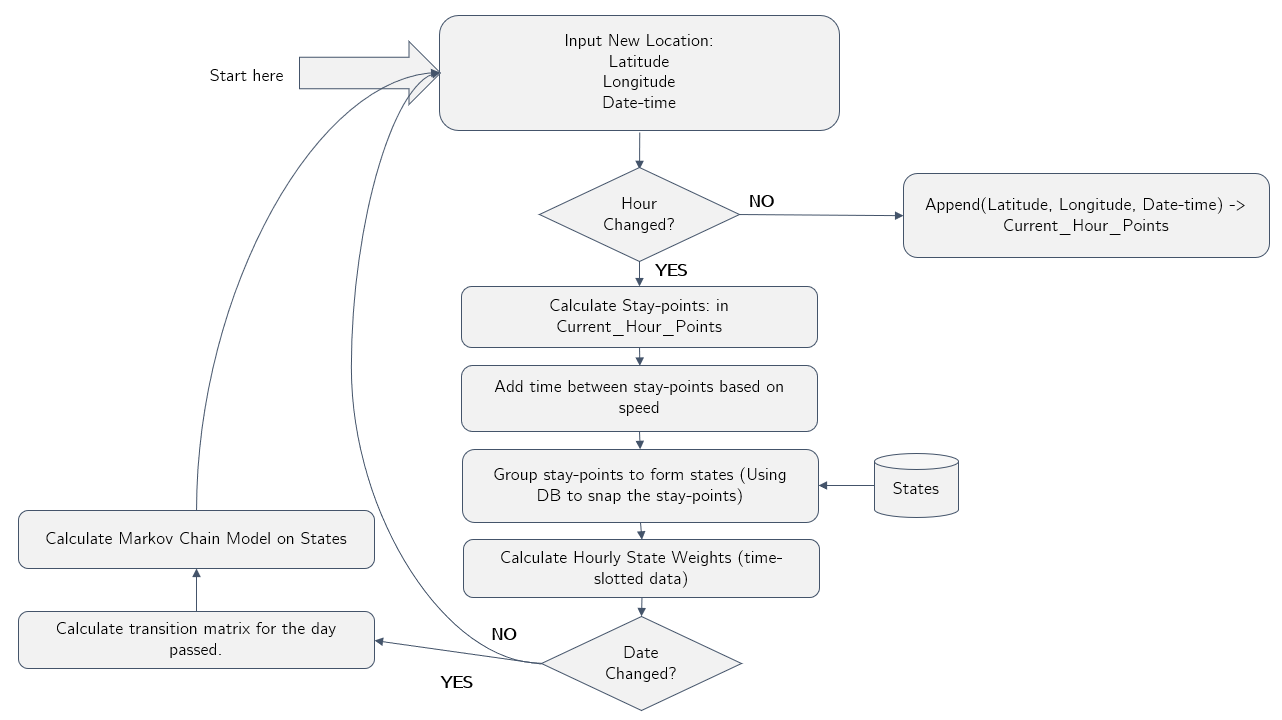


Figure 1: Design Flow-chart

This model helps predicting user future locations. We extend this location model to predict the future locations with confidence percentages.

# **Related Work**

Here we introduce the related work in field, the references and motivational work and few parts in details.

Baumann et al. compared 18 different location prediction algorithms. They focus on the accuracy of prediction along with other parameters to compare different algorithms. Based on their analysis, they also present a new next-place prediction algorithm called MAJOR. The dataset used by the researchers is Nokia Mobile Data Challenge (MDC). This data contains 37 user mobile phone data over 1.5 years.

They considered spatial and temporal features like current location of the user P1, current and previous location of the user P2, time of the day H, Day of the week D, weekday or weekend W. Here P1 and P2 are spatial features and H, D and W are temporal features. Using the combination of these spatial and temporal features, they have formed several algorithms e.g., DP1, WHP2 and so on. On these algorithms, they have calculated performance metrics which contains factors like accuracy percentage A1 which is the ratio of correct predictions to the total predictions. Other performance factors which included true positive, false positive, true negative and false negative with respect to transitions. For example, a true positive transition is the transition which is correctly predicted from one place to another and true positive transition rate TTPR is the ratio of true positive transitions over the total transitions. Like this they have also calculated false positive transition rate and so on. Some other interesting performance parameters included transition precision ratio which is calculated as ratio of number of correctly predicted transitions and the total predicted transitions. They also considered the arrival and departure events prediction from a particular place as a performance metric.

Using the combination of different spatial and temporal features, they compared 18 different prediction algorithms for their predefined performance metrics. The first comparison is highlighted for algorithms considering only spatial features or only temporal features or both together. Most algorithm who can achieve good prediction accuracy fail to predict a transition and vice versa. This led to a conclusion that there exists a trade-off between prediction accuracy and transition prediction. This is overcome by the novel approach introduced in the paper called MAJOR. This new approach run all 18 algorithms (spatial and temporal combinations) together and select the one with highest vote. This gave MAJOR an accuracy of 82% but only 21% detection of true transition. To improve the transition detection ability, they have introduced a voting threshold. The analyses suggested that a median of 8 approaches predict a true transition and a median of 3 approaches predict a transition when no transition occur. This will help deciding voting threshold offline. If the minimum number of approaches voting for the transition is greater than of equal to the threshold, then it is considered a transition, otherwise no.

Noulas et al. also have contributed in next place location prediction. The main idea is to use user check-ins on Foursquare to predict user movements. They have used 35 million check-ins from across the globe over the period of 5 years. They have explained how user check-ins not only allow us to see the locations user visited in the past but also help us understand the mobility patterns of the users. They have used the prediction features like user preferences and popularity of the places, geographic distance. On these features, they have used supervised learning linear model and M5 model trees.

One of the first tasks addressed are the next check-in prediction. They rank all the possible next location check-in based on current check-in and suggest that 99% of the next check-ins are within 10 kilometers radius from the current check-in and is mostly in urban areas. The ranking is performed based on historical visits by the user to a place, categorical preferences based on what category of places have user checked-in in past and social filtering based on what user friends have checked-in. The next task is a global mobility feature to determine check-in patterns irrespective of user preferences. This uses popularity of the geographic location, geographic and relative distance of all the other locations from user’s current location, activity transition where few locations are visited after specific locations, for instance going to a hotel after an airport or railway station visit. Next, they assign temporal feature to places. Based on the hour category, what type of places have been checked-in at a particular hour of the day or week.

After the assigning these features, they define the rank(k) for a venue, percentile rank (PR) and average percentile rank (APR). The analyses from the researchers suggests that APR is scored higher for categorial preferences higher i.e. 0.84 when compared to historical visits with APR 0.68 and social filtering with APR 0.61. In global mobility section, place popularity has better APR which is 0.86. Activity transition features also achieve only 0.60. The study also suggested that people tend to stick to their set of location check-ins during the day time but visited new locations during evenings. All these features suggest that there could be many factors which can affect user movement patterns. They finally used all these features and combined them into a supervised learning framework. With M5 tree, they have received an APR of 0.94 and linear regression model only resulted with an APR of 0.81 which is lesser than many individual feature APR.

Gomes et al. also discusses mobile based next location prediction based on current location. They suggest using contextual data along with spatial and temporal data associated with location. The mobile call/SMS logs, accelerometer and Bluetooth can have additional information which has not been investigated before for location predictions. The researchers explain how location prediction is very user specific, the data is evolving with changing city/work location etc. and it is possible to have missing location data.

The steps that the model performs are pre-processing the raw data which keeps the short-term data with its contextual information, the model should accept and integrate new location data and an updated check of actual next location vs predicted next location to keep an updated accuracy rate. This model is implemented in an online manner on a mobile device. The data used as input of from Nokia Mobile Data Challenge (MDC) from 200 participants over one year. First the raw data is processed to extract temporal, phone status, phone usage and other features. Then a classification technique is used with a software name WEKA.

The results suggested that the regular users are easy to predict with an accuracy percentage of 80% where as the users with irregular movement patterns are difficult to predict. In feature selection phase, it has been found that keep all the features gives the best results with accuracy of 92%.

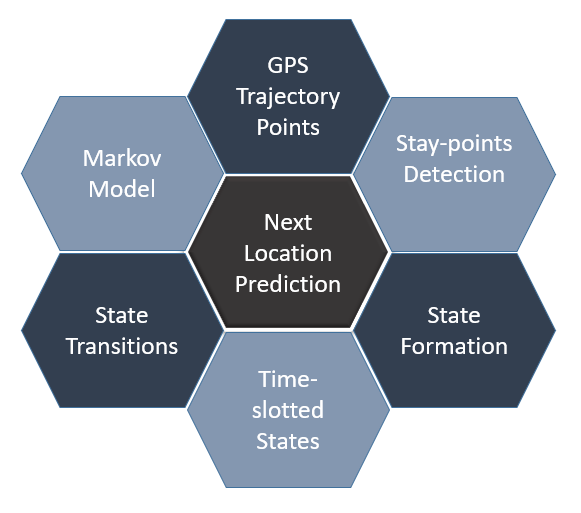
The paper also suggests an alternative advertisement approach. For instance, a user will be more interested in dinner promotion/discount before he/she goes outside on dinner time. This way, the telecom service provider can push the relevant advertainments on user phone based on the predictions without disclosing user’s information or location data to third parties. In this scenario, user can receive more relevant advertisements and still have not shared his private information with advertisement companies. This can help in preventing personal data to be shared with companies and third parties and hence preserving user privacy.

Baratchi et al. design hierarchal hidden semi-markov-model concerning spatio-temporal of location data to predict human mobility patterns. In this model, each state denotes either a stay-point or a transition from one place to another. The states which are more visited will be super states consisting of other states, and the states geographically closer or spatio-temporally closer are more likable to be in one state. Hence there are super states which contains other states. The next step is to map each location coordinate in a grid with cell id. The algorithm becomes expensive with increasing states. The states can be reduced by using states at higher level in hierarchy. This in turn, reduces the total complexity of the algorithm. The researchers have used real life data Geolife dataset and Capricorn dataset. The approach has better results in the presence of noise and missing data.

Jong Hee Kang et al. suggests that users are more interested in “places” rather than location. By “places” they mean where the user work/live/play or so on. They uses Place Lab to collect user location data from Wi-Fi enabled devices which works best also for indoors locations. They also introduce a time-based clustering algorithm to determine user locations. This algorithm waits for the next location to determine the significant user locations. The cluster within a distance threshold which is stayed for at least a given time threshold is considered as a significant place. Hence this clustering excludes all irrelevant or shortly stayed places. Their results show that the algorithm could extract significant places. The researchers also suggest that the locations must be labelled in order to extract semantic meaning behind the location coordinates like work/restaurants and so on.

# **System Model**

The system model for next location prediction model contain several steps. The process takes GPS trajectory points as input and process them to markov chain. The intermediate steps shown in figure are stay-point detection, state formation, time-slotted states, state transitions.



The human mobility pattern can be dependent on several features like user’s occupation. For instance, if the user is a sales person or has an occupation which requires daily travel, it is very unlikely that the user has a regular home-work-home pattern. These users are difficult to be predicted. There could be other users who have very regular movement patterns. These users are easily predicted. The idea is to have a prediction model which can work for everyone.

User tend to have a pattern where the next location is dependent on their current location. Consider an example where work is often visited directly after home, but home is not very often visited after work. This is a common trend where the user visits restaurants, gym or some other location after work before he/she comes back to home. Another example is that after supermarket visit, the user tends to go back home. These trends could be very often predictable but also sometimes not obvious. For instance, a restaurant visit could occur after home or work visit or even after a shopping mall visit. Hence, we can say it confidently that a large amount of movements is dependent on current location. Hence the thesis suggests predicting the next location based on markov chain which are build on states representing user significant places based on his visits.

## Components

In the model, the location data is input as an online GPS location data on a mobile system. This is to simulate how user share the location data with other Location Based applications like Google+, Facebook, etc. These locations are sampled and the noise (travelling locations or short stay locations like post-office visit) are removed. This makes sure that the markov model is built on stable and longer stayed locations which are later formed as states. These states can represent many different locations with different semantic meanings like home, work, favorite restaurant, gym/club, etc. These states are recorded for everyday and distributed on hourly basis. The hourly weighted or time slotted data is then used to form the markov chain.

The location coordinates are read from Geolife dataset user files in an online manner. This location data has latitude, longitude, date, time information along with some other information. This location coordinates are read in an online manner to extract the locations where user has spent more time. These locations are called as stay-points SP. The extracts of stay-points are stored as SP1, SP2, … SPn. These stay-points are the significant places for this user which has semantic meaning behind the location coordinates specific to this user.

* Once the stay-point is extracted, we snap these stay-points to states which forms S = {st1, st2, …stn}. If there exist a state for the stay-point location, it is snapped to the existing state. If a new location stay-point is found, a new state is formed. These states are the unique list of stay-points for the user at any time. Once we have the states, we create time slotted states for each hour as shown in the figure 2.

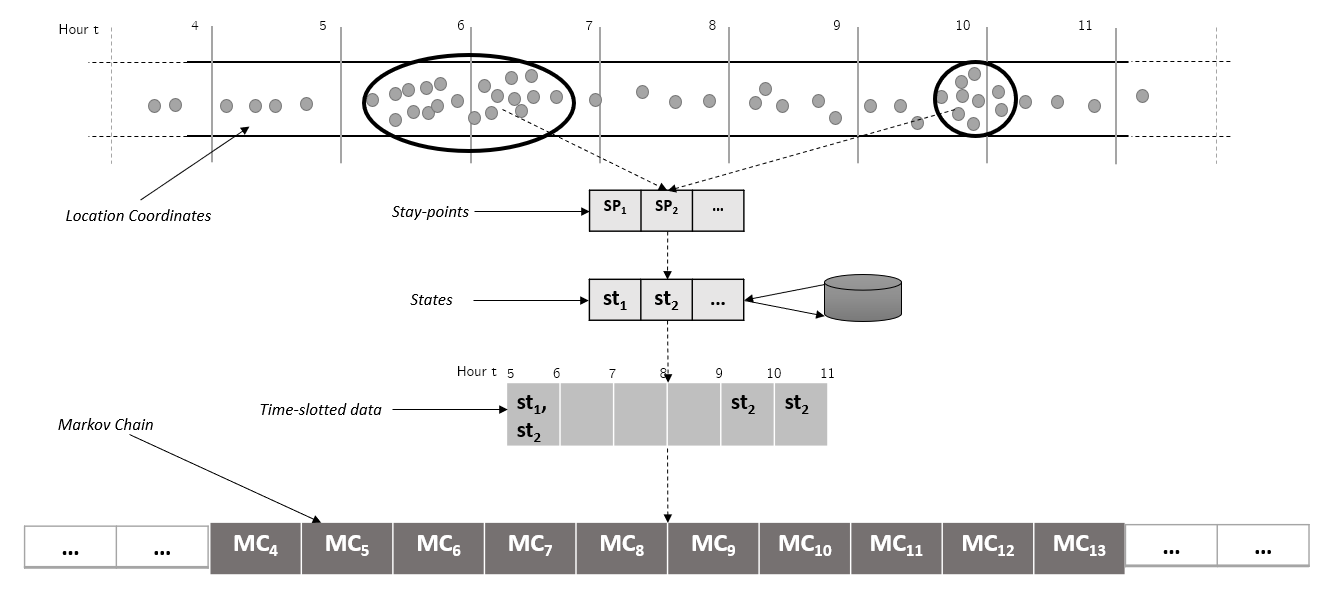


Figure 2: GPS coordinates snapping

Based on these states, we form time-slotted markov chain on the states. The markov chain holds the probability from st1 to all the other states at this time slot.

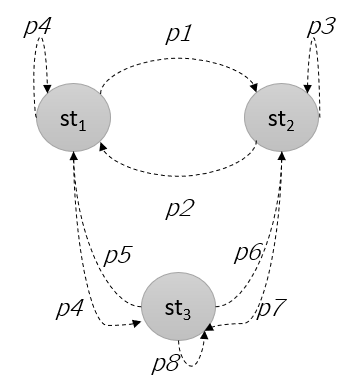
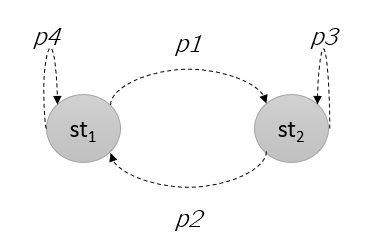


Figure 3: Markov chain on states

This markov chain model is then used later to predict the movement from st1 to all the other states at any given time-slot. The similar probabilities are calculated every time a new state is added as depicted in figure 3 for a new state st3. This model expands as the number of states increases and so does the complexity of the markov chain and location prediction.

## Assumptions

Occurrences of few popular locations like home and work for a user will be more compared to other location. This helps in marking the home and work location while analyzing the data. The home location is often the one which has occurrences during midnight and on weekends or vacation days. Another assumption is that the location is turned on for most time of the day.

However, the location data is sometimes not available to be shared. For instance, there is no internet in a skyscraper work location and the location is turned off as soon as the user has entered the work location. The next location input is after few hours from work location again while user comes in a network coverage area. The first and the last known location helps us to fill the missed information during the few hours based on the distance and time difference between the two locations.

## Problem Statement

Use of mobile devices is ubiquitous. Users are often using applications on the mobile system which uses their location coordinate. Although the location usage is consensual, the duration of sharing is often not well understood by the users. The applications use the location data in the background when the application is not even in use. According to a Pew survey, 91% of Americans agree that they have lost control on their private data and 64% reported that government must regulate advertisers. Another survey reported that only 9% users are confident that social media companies will protect their data.

The next locations can be predicted based on the user’s current location. The prediction can be done for next hours based on this information shared. For instance, if the user who usually leaves for work at 8 am and stays till 6 pm at work. Here it can suggest if the user is at work at 8 am, he will be at work till 6 pm. The location coordinates at 8 am were enough to predict his/her location till 6 pm. The threat of location prediction and exploitation of user privacy is to be shown to users so that the user can make a wise decision before sharing the location with third party applications on mobile devices.

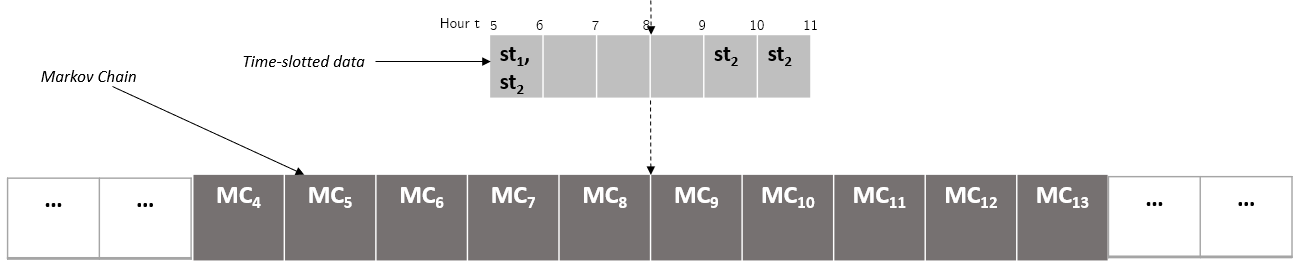
The idea is to simplify the location prediction algorithm on python and test the prediction accuracy. The same algorithm is to be implemented on a mobile device. The model of location prediction is to be built which takes in the location coordinates as input in an online manner and forms a markov chain model. This markov chain model must be then used to make predictions based on current location and hour of the day. The location prediction should continue till the confidence falls below a threshold value. This should be shown to the user as what predictions can be done based on a known location at a particular hour.

# **Markov Chain Model**

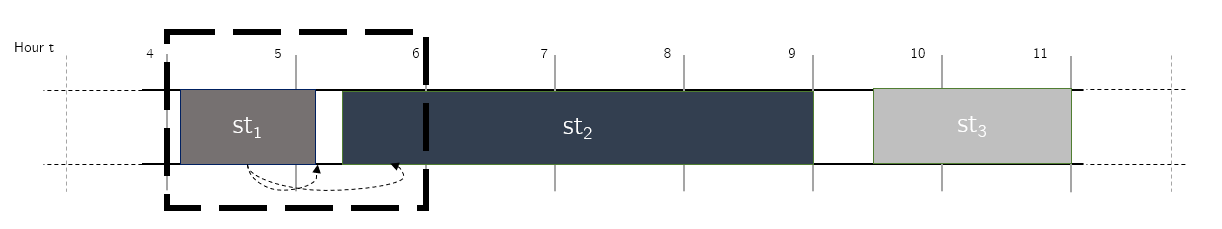
## 4.1 Markov Model for Location Prediction

The markov model for location prediction is formed based on states. The assumption in this section is that the states are already created from location. The state formation from raw location coordinates is explained in further sections.

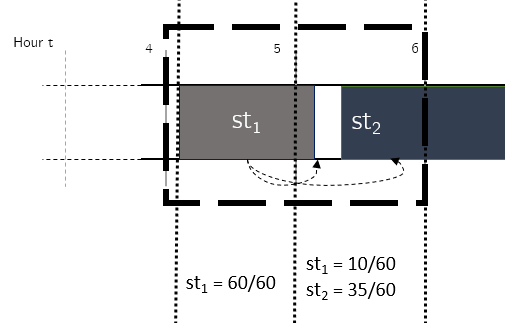
The states are the significant locations for a user extracted from raw location coordinates. The symbolic meaning behind these states are, for instance, home, work or gym. The markov chain holds the probability of transitioning from one state to another. The transitions are recorded for each hour of the day as shown in the figure.



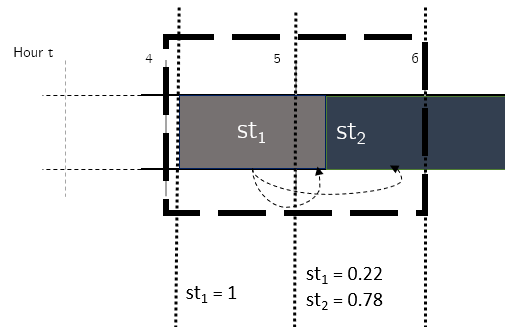
The markov chain model is formed from time-slotted states data. Consider an example as shown below in the figure. There are three states st1, st2 and st3 which exists between hour 4-6, 5-9 and 9-11 respectively.



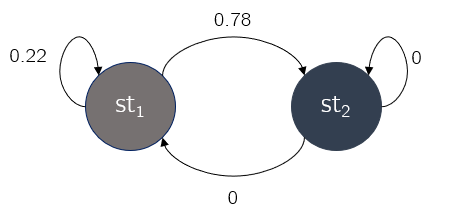
The transition from hour 4 to hour 5 is to be build. In this example, the state transition from hour 4 to hour 5 is from st1 to st1 and st1 to st2. It is important to mention that the hourly weights are normalized before the markov chain is calculated.



After normalization of hourly weights, the states will be filled with states as shown in the figure.

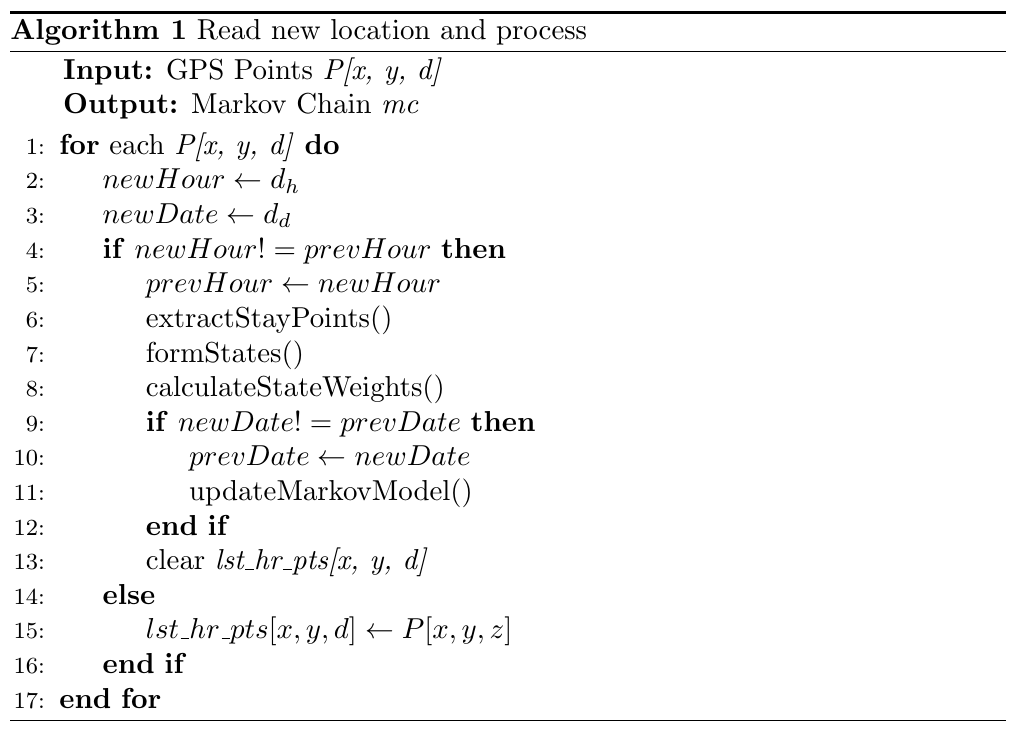


The probability of transitioning from st1 to st1 is calculated based on the hourly weight of st1 in hour 4 and hourly weight of st1 in hour 5. Similarly, the probability of transitioning from st1 to st2 is calculated based on the hourly weight of st1 in hour 4 and hourly weight of st2 in hour 5. The markov chain for transition from hour 4 to hour 5 is called MC4 which is depicted in the figure.



Using the markov chain, the predictions are done. The predictions are based on user’s current location. The probability of going to all the other places at this time-slot is from current location is already present in markov chain. The probability is also used to define the confidence of the prediction.

The GPS trajectory points are received as input. For each new GPS Point P[x, y, d], which contains x as latitude, y as longitude and d as datetime, the process is run. The first step is to detect the time-slot change. For every hour, the extraction of stay-points extractStaypoints() is run giving the sp = {sp1, sp2,… spn}. From these stay-points SP the states st = {st1, st2, … stn} are formed using formStates(). These states are used later for markov chain model. The states’ S represent “home”, “work” and other important visited places. The next step after state formation is the calculate state weights w = {w1, w2, … wk} in this time-slot. Once the day is changes, the markov model is created mc using the state weights w. The individual algorithms of stay-point detection, state formation, state weight calculation and markov chain model creation is explained in detail in chapter 5.



# **Implementation**

## 5.1 Variables Used

The table below is to provide an overview of the variables used in the further sub-sections. The variables are used in algorithms and in explanation. The list of variables covers majority of variables used in further sections, but it is not exhaustive. Few new variables are introduced in the further section for clear understanding of concepts.

|  |  |
| --- | --- |
| Variable | Description |
| P[x, y, d] | point is a tuple:  (Latitude, Longitude, Datetime) |
| dh | Hour from d datetime |
| dd | Date from d datetime |
| th\_tck | Threshold time for tracking GPS location data |
| th\_d | Threshold distance for staypoints |
| th\_t | Threshold time for staypoints |
| spi(x, y, ds, de) | ith Staypoint  (Staypoint Latitude, Stapoint Longitude, Start Datetime, End Datetime) |
| sti(x, y) | ith State  (State Latitude, State Longitude) |
| lst\_hr\_pts(x, y, d) | Last hour GPS points  (Latitude, Longitude, Datetime) |
| w | State hour weights |
| mc | Markov Chain for sti to sti+1 for time slot h to h+1 |

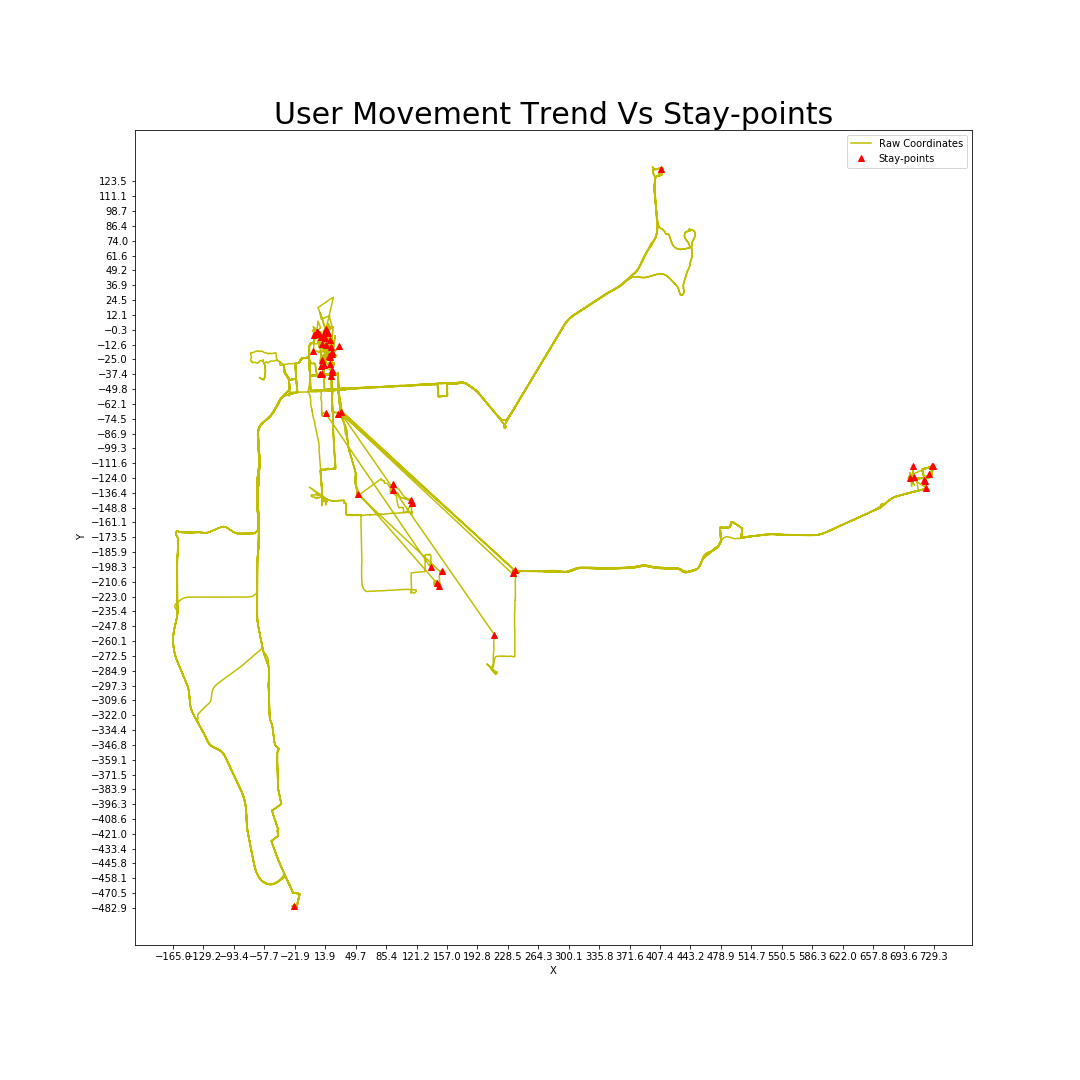
## 5.2 Stay-points

Stay-points are those significant places where user spent significant time. Stay-points are any points which are stayed by the user in user trajectories or it is the start or the end of the trajectory. For example, if user start at his home, the home itself is a stay-point. Now he moves towards work, but he visits a cafe in between for breakfast. The cafe is also, a stay-point and then he finishes his trajectory at work, where work is again a stay-point.

The places like cafe in this case is identified using distance and time-based clustering. Distance and time-based clustering work best in case of location data. This clustering the not so complex and can be run on a mobile device as a background process. The clustering has two thresholds, one for distance (th\_d) and one for time (th\_t). These threshold help determining the stay-points in an online fashion. The location points within the radius of distance threshold (th\_d) with time spent at this location greater than or equal to the time threshold (th\_t) is regarded as a stay-point. For example, a set of points within 200m of radius with total duration of stay greater than 20 minutes can be regarded as a stay-point.

Trajectories are continuously received GPS points. The gap of time greater than tracking time threshold (th\_tck) between two GPS points breaks the old trajectory and starts a new one. Note that the stay-points are found within the trajectory with this algorithm. The other type of stay-points is where user has ended or started his trajectory. For instance, the user has entered his work location and now we don’t share hi location for a threshold tracking time. This means, if there have been no new location coordinates received for a given time, the last point is added as a stay-point and so is the next point received consequently.

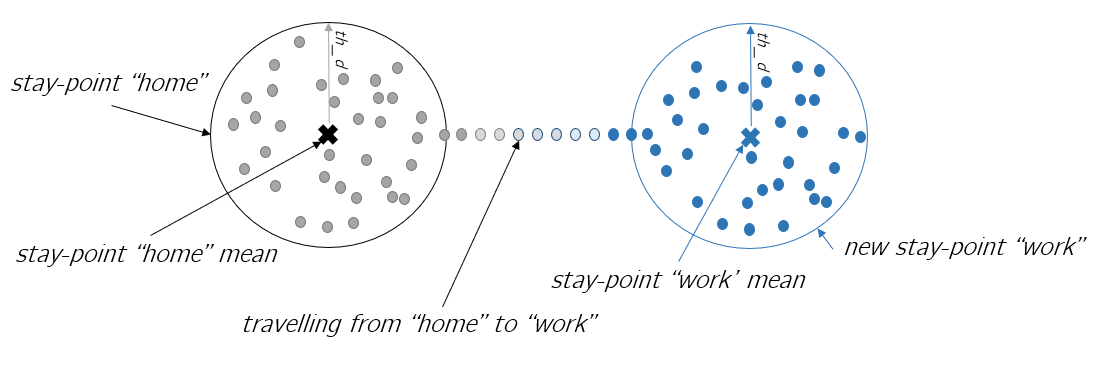
After the collection of stay-points, the stay-points entering and leaving time is changed. This is done to estimate the time of leaving a stay-point or time of entering a stay-point. For instance, user is reported to be at home at 7 am and then the next stay-point is found to be work at 8 am. The missing data between 7 am and 8 am can be for many reasons, for instance, no network coverage or user has purposefully turned off the location sharing. If the distance between these two locations is X kms. This information helps us to estimate the leaving time from home and arriving time at work. The speed of user (spd) can be calculated as X kms / time minutes i.e. X/60 kms/mins. Now, the estimated time for leaving home is calculated as 7 am + min (th\_d, spd/60)/spd. The similar calculation is done to estimate the time of arrival at work.



The figure depicts the stay-points for user 0 for April 2009. The yellow line shows the user movement from raw trajectory and the red markers exhibit the stay-points extracted. This clearly depicts that a lot of noise in the trajectory data is removed at this step and only the significant stays are extracted.

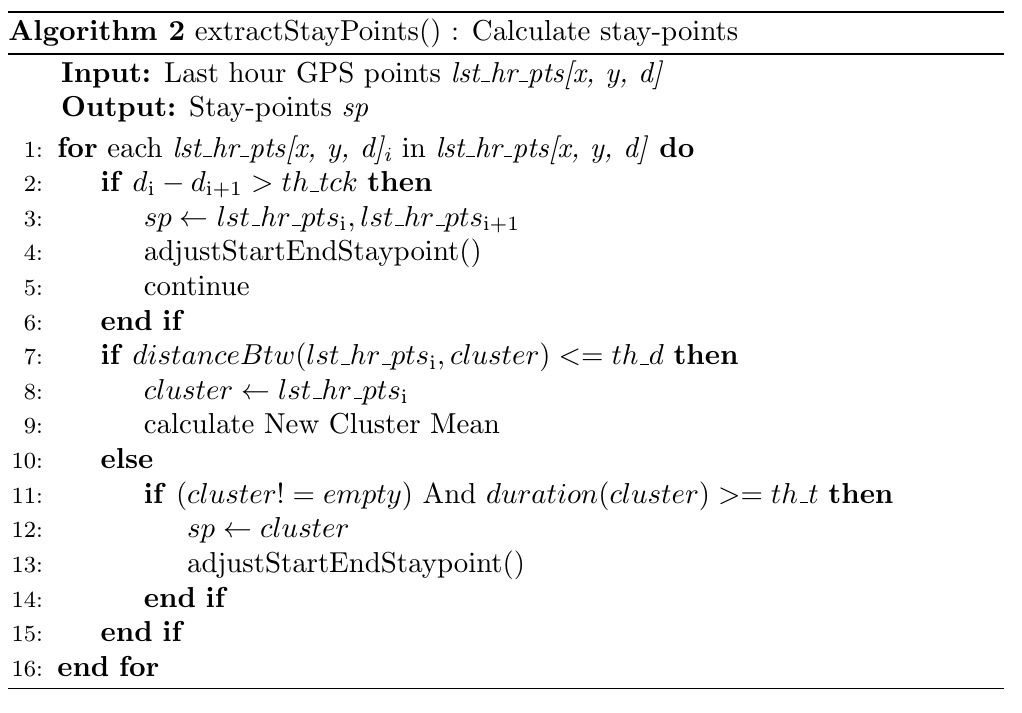
### 5.2.1 Algorithm

The stay-points are extracted from raw points to remove the noisy points. The noisy points could be travelling with the bus or train or a short stop at the letter box. The stay-point extraction is the process extracting longer stayed locations from raw GPS trajectories. The figure shows the transition from “home” to “work”. In this case, both “home” and “work” are extracted as stay-points.



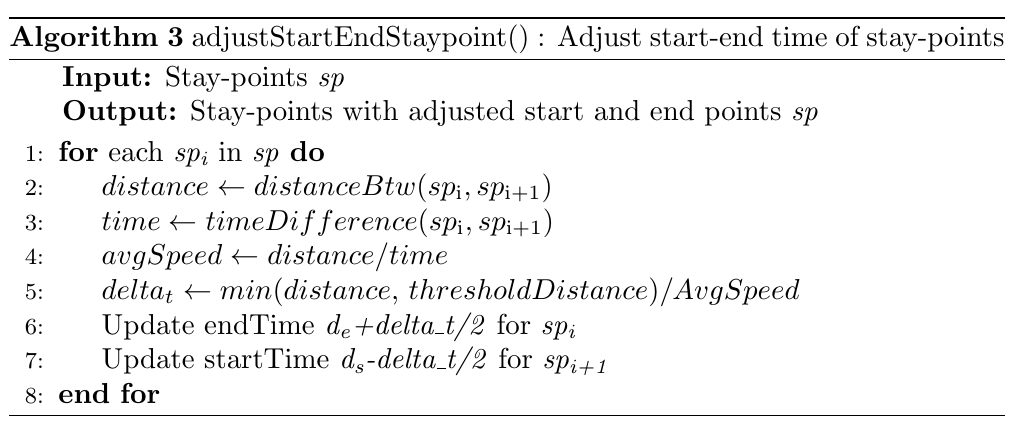
The extraction of stay-points takes lst\_hr\_pts as input and generates sp = {sp1, sp2,… spn} as output. The algorithm cluster the points within the radius of stay-point distance threshold th\_d for a minimum duration of time th\_t. The selection of distance and time threshold is very important. If the distance threshold value is too large, the mean of the stay-point locations will be a confusing location on the map. If the time threshold value is very small, a lot of insignificant locations will be added as stay-points.

A new location from lst\_hr\_pts is added to the cluster if the distance between the new point and the cluster mean is less than or equal to the distance threshold th\_d. Every time a new point is added to the cluster, a new mean of the cluster is calculated and the process repeats. If the new point from lst\_hr\_pts is moving away from the cluster mean, then the point is not added to the same cluster. This means that if the distance between the mean of the cluster and the new point from lst\_hr\_pts is greater than threshold th\_d, then the new point is not added to the cluster. At this point, the cluster duration is checked. The cluster duration is nothing but the largest datetime – smallest datetime from the cluster elements. If the cluster duration is greater than or equal to th\_t, then the cluster is added as the stay-point sp with latitude and longitude as cluster mean. The new point from lst\_hr\_pts is also added as a stay-point if the difference of time between the new point and the previous point is greater than time tracking threshold th\_tck. This is to ensure that if the GPS points are not received for a long time, we add the last and the new point as a stay-point assuming the end of the previous trajectory and the start of the new trajectory.



The stay-points are often not continuously distributed over time. Consider a scenario where a user is at “work” location till 9 am. After entering work, either user decides not to share the location or enters a no-network coverage area. The next stay-point detected is “work” location at 1 pm when user came out for lunch from the building. The time of stay at “work” in this case till 9 am is misleading as user stayed at this location till 1 pm. Hence, once the stay-points are collected, we adjust the starting time and the leaving time of each stay-point.

This is done by comparing each stay-point in sp = {sp1, sp2,… spn} to it’s very next stay-point in sp. Now, the distance and time difference between the two stay-points spi and spi+1 is calculated. Using this the average speed of travel can be easily calculated which is distance/time. The delta time delta\_t is calculated as division of minimum of distance between spi and spi+1 to the average speed. Now we add the delta time delta\_t to spi to change leaving time at the spi location and subtract delta time delta\_t for spi+1 to change the entering time at location spi+1.

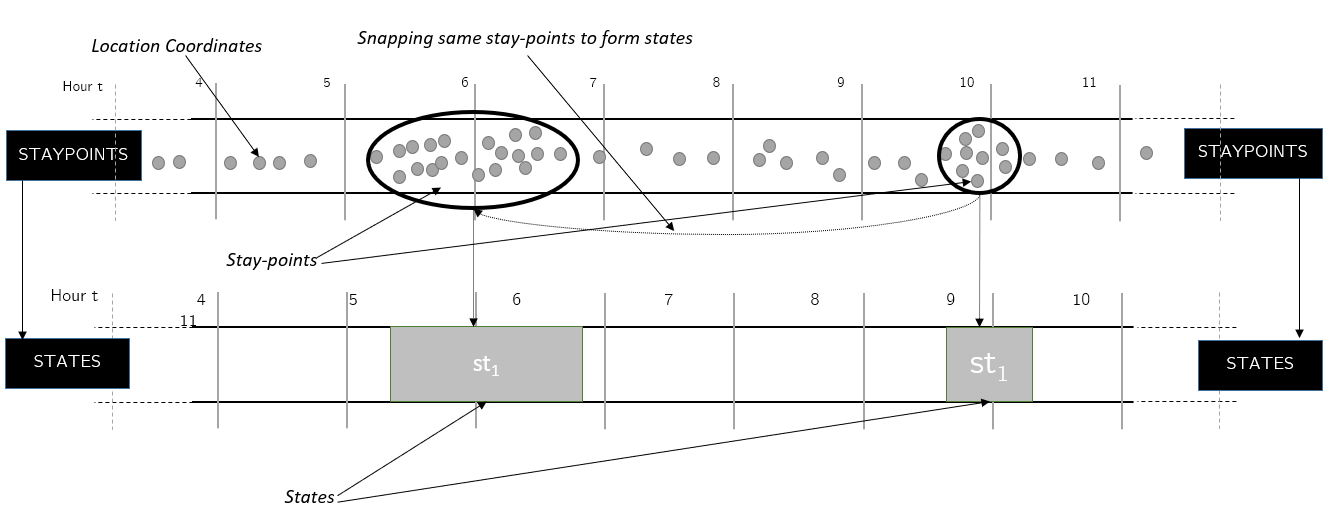


### 5.2.2 Implementation Result

The algorithm is applied to Geolife dataset. The user files are read in an online manner to simulate the GPS location points received on a mobile device.

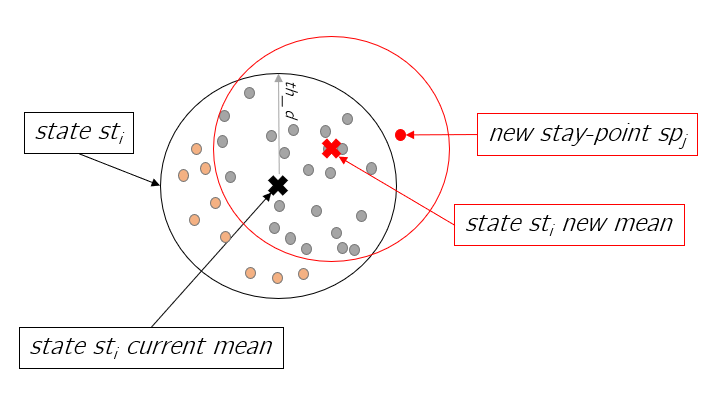
## 5.3 States Formation

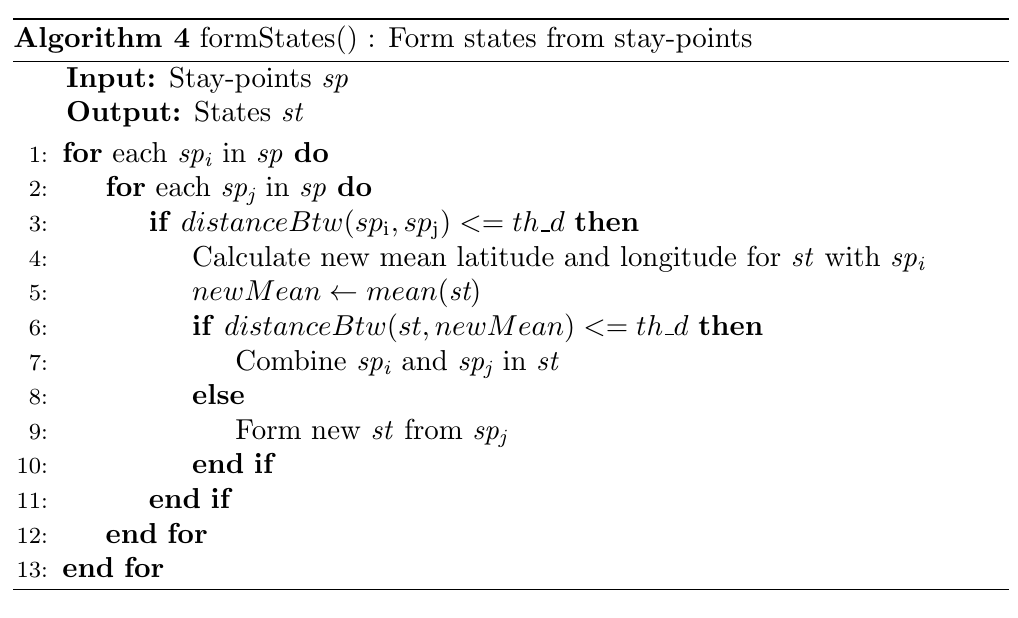
A state is formed using a group of stay-points represented as S = {st1, st2, …stn}. Each state has a semantic meaning like “gym”, “restaurant”, “home” or so on. The states are found using a distance threshold for states. The states are unique list of stay-points that are found so far. All the stay-points within this threshold distance radius is grouped together as a single state. This is called snapping stay-points to the states. This is depicted in the figure below. The stay-points are snapped to states to form st1. The mean of all stay-points forming the state is stored. Every time a new stay-point is snapped to the existing state, a check is done if all the existing stay-points remain within the radius from the state mean center. If the new point to be added is within the radius of threshold distance but the new mean center of the state throws out the already existing states, then a new state is formed. Hence, a new stay-point is only added to the state if after calculating the mean of the new state, all the existing stay-points still stay within the distance threshold radius from the mean center of the state. This is done to avoid drifting problem while aggregating the stay-points into states. Finally, markov Chain model is applied to the states.



### 5.3.1 Algorithm

Once the stay-points sp are extracted for this time slot, the states are formed. Each stay-point in sp = {sp1, sp2,… spn} is compared with every other stay-point in sp. If the distance between the two stay-points is less than the distance threshold th\_d, then the two stay-points are combined to form a state st. The stay-points are combined with an exception. The figure below depicts the drifting problem. The current mean of state sti is marked with **×**. The addition of the new stay-point spj will make the mean of the state shifted denoted by **×**. The new mean of the state sti will throw some of the existing points from left out of the state radius. Hence the new stay-point spj is added to a new state in this case. Hence, the idea is, while adding the new stay-point to an existing state, a check is done. If all the existing stay-points stays within the radius of the new state mean, then the stay-point is snapped to the state st, otherwise a new state is formed. In other words, if any of the existing stay-points contributing to the state formation is moving out of the radius, then the new stay-point is not added to this state and a new state is formed. This is done to avoid the drifting mean problem.

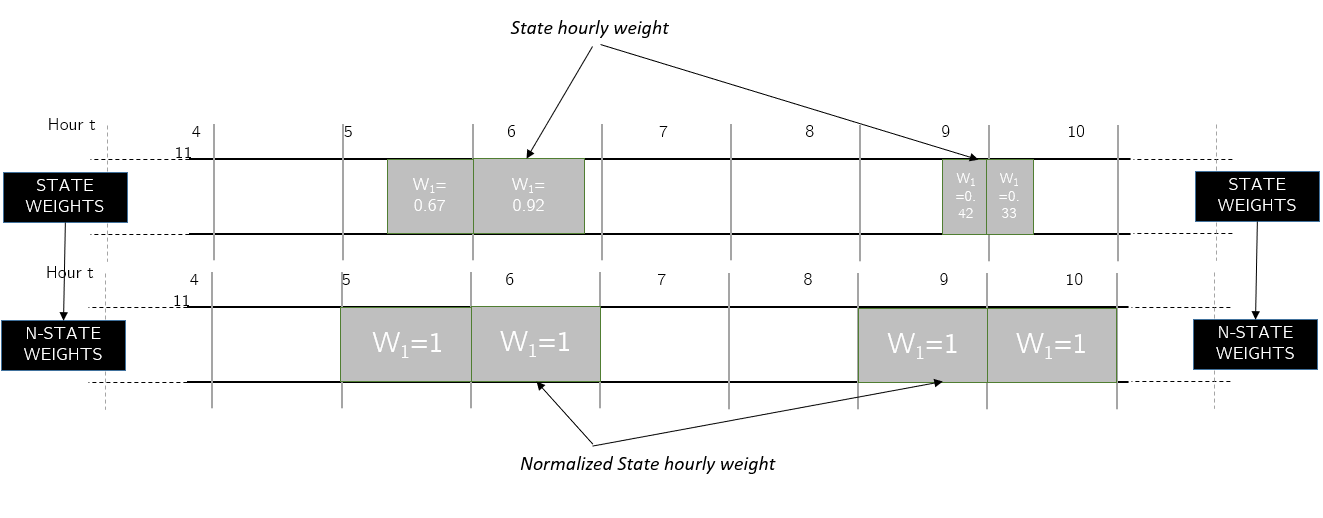




### 5.3.2 Implementation Result

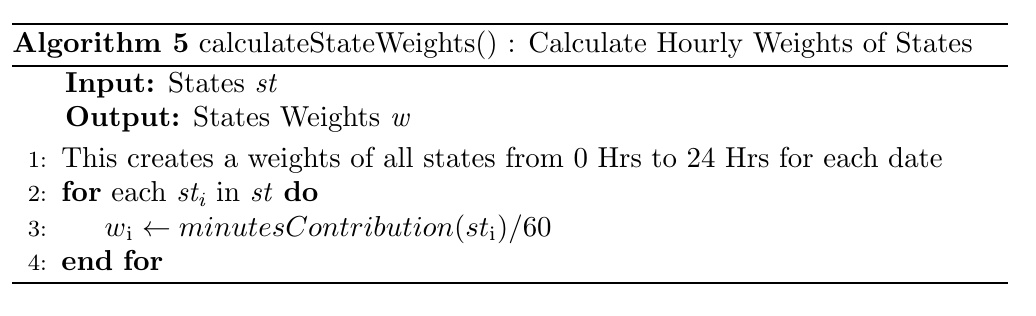
## 5.4 State Weights

The hourly weights form the time-slotted state data. The states can appear in different time-slots. The set of states S = {st1, st2, …stn} are assigned with weights at each time-slot to form Wt = {w1, w2, …stn} for timeslot t.The hourly weights are the ratio of minutes spent at a particular state to the total minutes in the hour. The weights are then normalized to smoothen the data in each hour slot. As shown in the figure, the ST1 hourly weight is calculated for 5-6, 6-7, 8-9 and 9-10. After the hourly weights are calculated for the state, the weights are normalized in each hour before they are used for markov chain calculation.



### 5.4.1 Algorithm

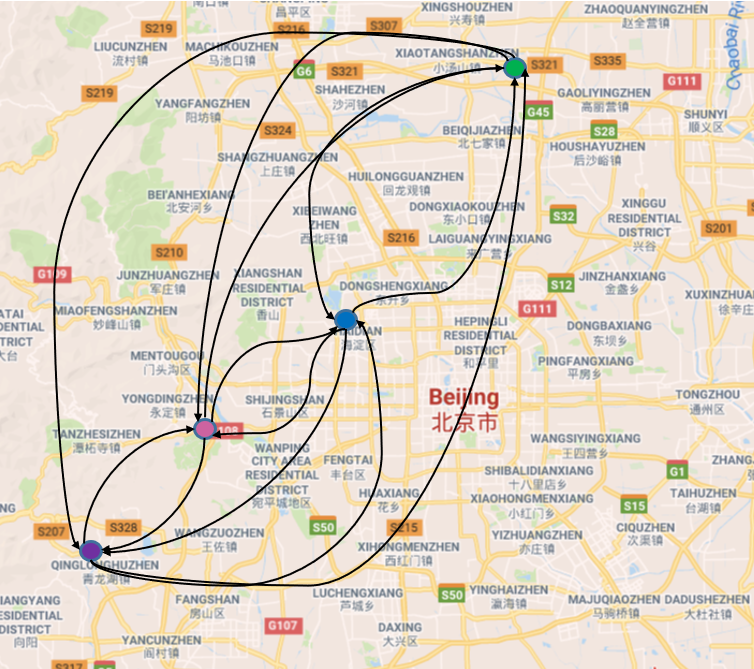
Once the states st = {st1, st2, …stn} are formed, the next step is to calculate the weight w = {w1, w2, … wk} of each state in this time-slot. It is important to know that there can be many states in one time slot. The weights w represents the probability of a state in this time-slot. Since states itself carry some semantic meaning like “home”, “work” or others, the weights w represents how long did the user stayed at “home” in this time-slot. Since in our case, the time-slots are divided into each hour, this means time slot t to t+1 has one-hour increment. The algorithm reads each state in st and calculates the weight in this time-slot by simply dividing the minutes contribution of each state with 60 minutes.



### 5.4.2 Implementation Result

## 5.5 Forming Markov chain

Once the time-slotted data is created and normalized, the markov chain is build. The model of markov chain is created to calculate the probability of going from one location to another. The figure shows on a map how the markov chain looks like.



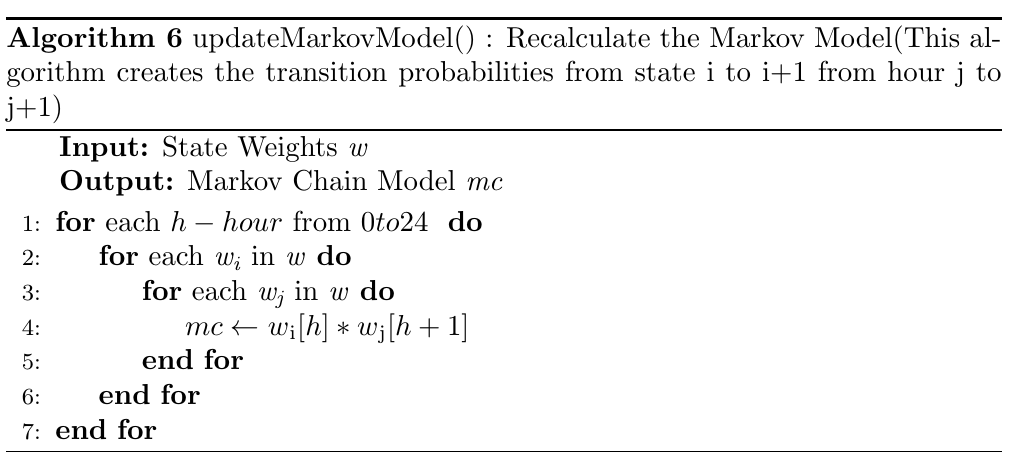
A set of n states’ S = {st1, st2, …stn} where each state represents a significant place or place of interest, with weights Wt = {w1, w2, …stn} at hour t and with weights Wt+1 = {w1, w2, …stn}t+1 at hour t+1 is used. The weight vector Wt and Wt+1 are multiplied to calculate the transition probabilities from each state in set S to every other state in S from time slot t to t+1. This transition probability is a theoretical measure to determine the chances of transiting from one place to another from one time slot to another. A transition can be a self-transition, for instance, st1 to st1 or a transition to another state, for instance st1 to st2. These transitions represent human mobility from one important place like “work” to another like “gym” for every time slot.

The markov chain built is standard and hence the probability only depends on the current location and not the locations before. This means a transition from “home” to “work” from time slot t to t+1 depends only on time slot t and not on any time slots before t.

### 5.5.1 Algorithm

After state weights w = {w1, w2, … wk} are calculated, the next step is to build the markov chain model. The model is updates after each day to keep a track of all the new locations user has visited in the previous day and update the mobility pattern. The continuous update of the markov chain model helps to track the changes in user behavior. For instance, user may change the work location or move to a new city.

The markov model mc contains the probability of transitioning from one state sti to another state stj for time slot t to t+1. This information is stored for each state in st = {st1, st2, …stn} transitioning to every state in st for all time slots. The time-slots in this case represents the hour of the day. For each hour of the day, the weights wi is multiplied with the weight wj for hour h to h+1. It is important to understand that a transition can be self-transition i.e. moving from sti to sti to another state stj.



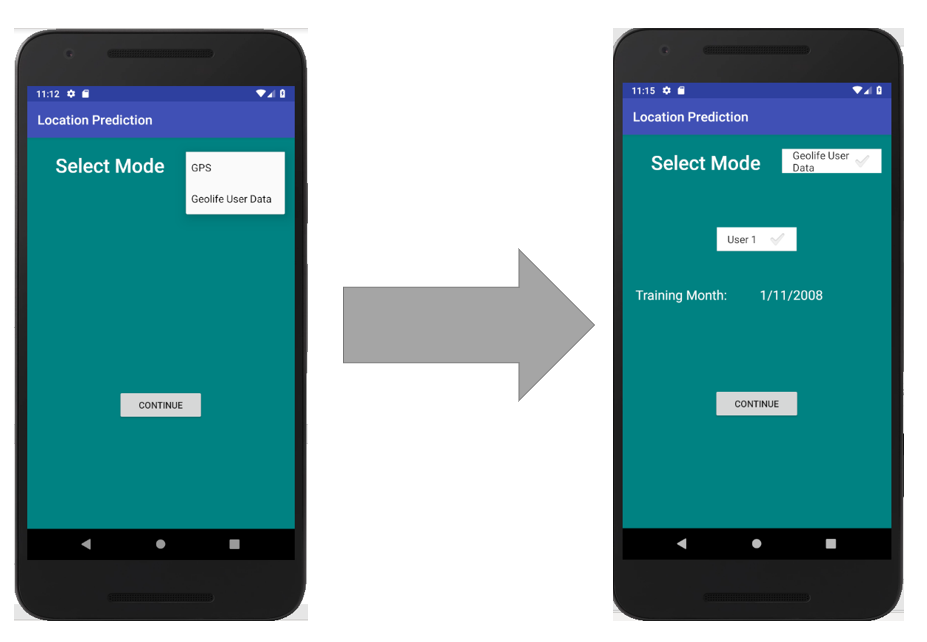
### 5.5.2 Implementation Result

## 5.6 Android implementation

The implementation on an android phone as a prototype. This is to showcase how the mobile devices receiving location data can exploit the data and make predictions. The markov chain algorithm is the same as explained earlier. The data is again from Geolife for prototyping purpose. In the actual scenario, this data will be fed from GPS to the application.

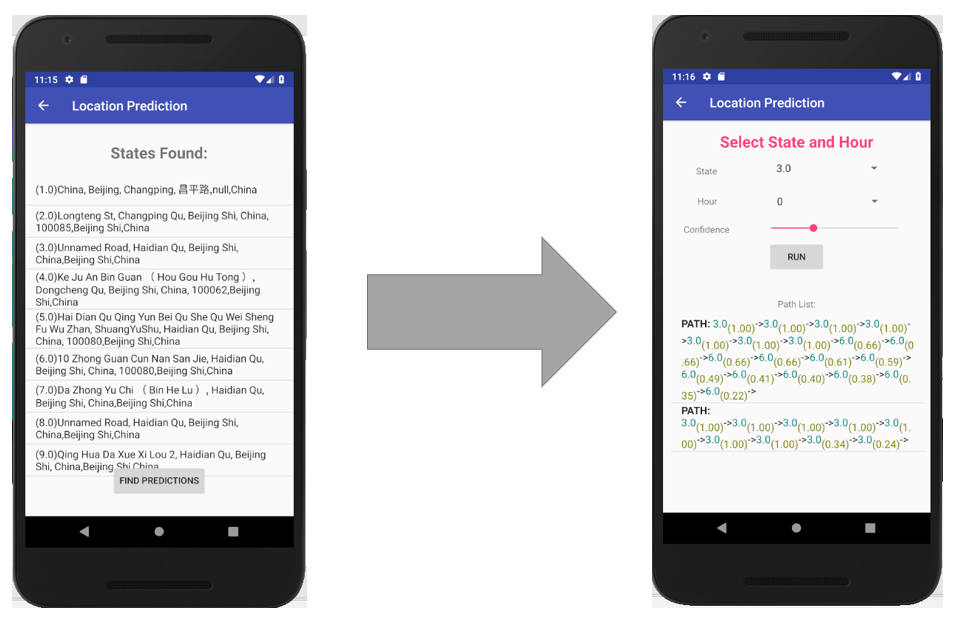
The android implementation has challenges like computational cost. The implementation is similar as the python implementation. The user can see the prediction results of sharing a location at a particular time.

The start screen of the android application has two modes “GPS” or “Geolife User Data”. Once the user has selected “Geolife User Data”, the option of choosing user and month is made visible. The user and month has to be selected from this input screen.



Once the user has made the selection and hit “Continue” button, the markov chain is build in the background from the data file. The states that are extracted which represents the significant places for the user are displayed as shown in the figure. The states may not be the exact positions but will be on the vicinity as they represent the mean of several coordinates within a range. The list is displayed to show the user all the visited placed that has been tracked down as important places.

Once the user proceed by pressing the button “Find Predictions”, user is taken to the final prediction scree. Here he/she can make a selection from the list of states and hour from the screen. The selection is to prototype if the user’s location was found to be at this selected state location at this hour, which paths could be predicted. The confidence level input can be changed using the slider. The paths are displayed below as a list with their corresponding confidence as a subscript.

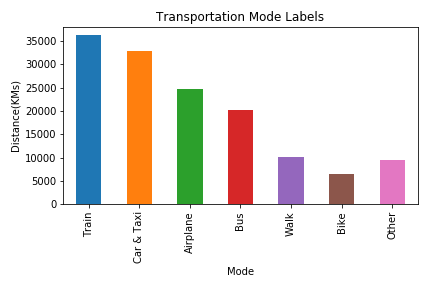


# **6 Evaluation**

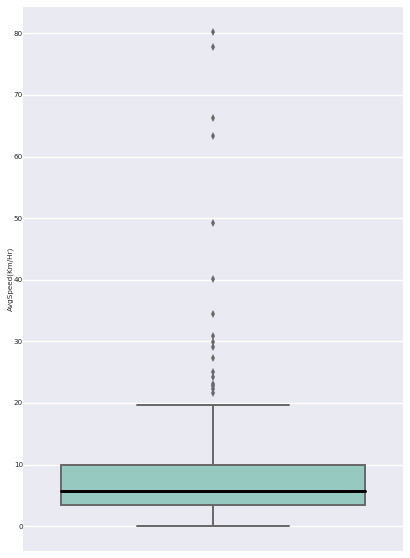
## 6.1 User data analysis

The data used for this master thesis is Geolife dataset from Microsoft. The dataset contains 182 users’ GPS trajectory data for the period of five years. The trajectory data contains the latitude, longitude, date, time, altitude information which is tracked every 1 to 5 seconds. Majority of the dataset was created in China with few exceptions of USA and Europe. This included several different types of users, few with a lot of trajectory data over years and few only for few weeks. For instance, user 17 has 1026179 trajectory points and users 72 has only 81 trajectory.

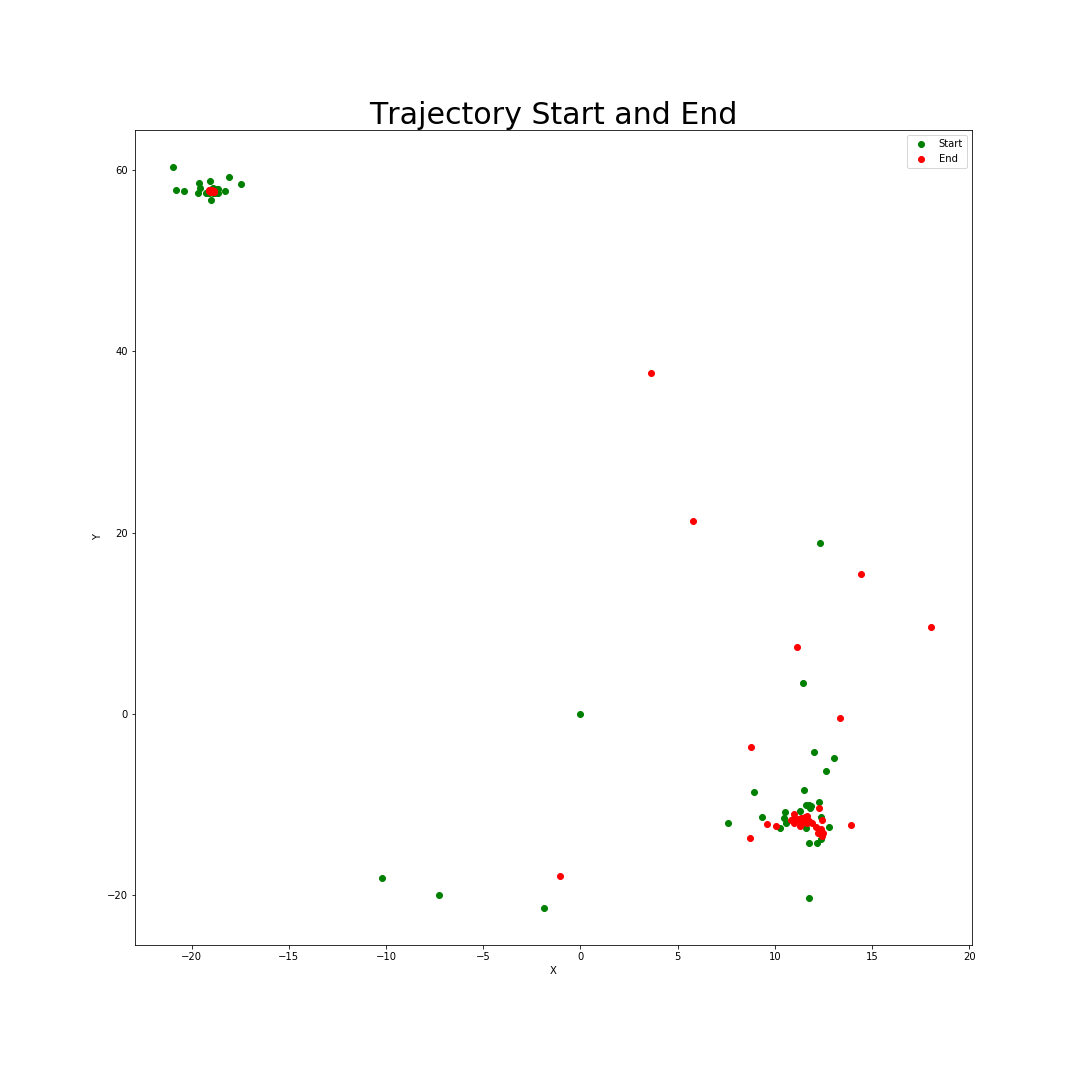
A total of 73 users have also labelled their transportation mode while recording the GPS trajectories. The figure below explains the distance travelled altogether using different transportation modes.



The transportation labels summary helped us to understand that most users were travelling while recording their GPS trajectories. The average speed which is calculated per trajectory file for each user in the dataset depicted the in figure below. The median travel speed was found to be 5.73 km/h. This indicates that most user tracked the GPS trajectories outdoor.



The analysis of each user also shows some important patterns. For instance, the first and last GPS coordinates of each trajectory is depicted in the figure. The green dots indicating the start of each trajectory and the red dots indicting the end of each trajectory. This also shows a pattern indicating that the trajectory data was mostly recorded outdoor. For example, the trajectory was always starting at home and ending at work, and then starting at work and ending at home.



## 6.2 Discussion and Summary

# **7 Conclusion and Future Work**

## 7.1 Summary

In this thesis, we developed an algorithm for location prediction using markov chain model. The dataset used is from Microsoft Geolife data. The prediction model is to simulate the privacy risks for mobile device users. The algorithm is first implemented on python to check the algorithm’s accuracy. It has been found that on an average of cosine similarity \_\_ can be achieved for future location predictions. The same algorithm is also implemented on Android. The result is shown to the user in which each future location is predicted with a confidence percentage.

## 7.2 Future Work