Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By SAVITHA BASKARAN

Entitled
VISUALIZATION OF SPATIO-TEMPORAL DATA IN TWO DIMENSIONAL SPACE

For the degree of Master of Science

Is approved by the final examining committee:

SHIAOFEN FANG
Chair

JAMES H. HILL

RAJE R. RAJEEV

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): SHIAOFEN FANG

Approved by: SHIAOFEN FANG 11/15/2016
Head of the Departmental Graduate Program Date

VISUALIZATION OF SPATIO-TEMPORAL DATA IN TWO DIMENSIONAL SPACE

A Thesis

Submitted to the Faculty

of

Purdue University

by

Savitha Baskaran

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2016

Purdue University

Indianapolis, Indiana

To my loving

husband, parents and brother

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Shiaofen Fang for his support, constant motivation and guidance during my study and research at the IUPUI. His trust and belief in my talent has led to the successful completion of this thesis. It was a great privilege for me to work with him. I would like to thank my thesis committee members: Dr. Rajeev Raje and Dr. James Hill. I would like to thank my husband who has been very supportive for the past two years and been my backbone of strength at the every obstacle I have faced during my Masters education. This thesis would have not happened without his constant motivation and inspiration.

My Parents have helped and supported me throughout my life and I wish to thank them for their love and encouragement. My little brother has been very supportive through my entire life and helped me with his constant cheering and pushed me further to succeed. I would also like to thank my friends and members of the Visualization Laboratory who have made my research at the IUPUI a enjoyable and memorable one.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

HSV	Hue Saturation Value
RGB	Red green blue
HTML	Hyper Text Markup Language
API	Application Programming Interface
Div	Division
GIS	Graphical Information System
GPS	Global Positioning System

ABSTRACT

Baskaran, Savitha. M.S., Purdue University, December 2016. Visualization Of Spatio-Temporal Data In Two Dimensional Space. Major Professor: Shiaofen Fang.

Spatio-temporal data is becoming very popular in the recent times, as there are large number of datasets that collect both location and temporal information in the real time. The main challenge is that extracting useful insights from such large data set is extremely complex and laborious. In this thesis, we have proposed a novel 2D technique to visualize the spatio-temporal big data. The visualization of the combined interaction between the spatial and temporal data is of high importance to uncover the insights and identify the trends within the data.

Maps have been a successful way to represent the spatial information. Additionally, in this work, colors are used to represent the temporal data. Every data point has the time information which is converted into relevant color, based on the HSV color model. The variation in the time is represented by transition from one color to another and hence provide smooth interpolation. The proposed solution will help the user to quickly understand the data and gain insights.

1. INTRODUCTION

1.1 Overview Of Spatio-Temporal Data

Spatio-temporal data is becoming very popular in the recent times, as there are large number of datasets that stores both location as well as temporal characteristics in the real time. Spatio-temporal data is a type of big data that contains attributes of both spatial and time related information. The main challenge is that extracting useful insights from such large datasets is becoming extremely complex and laborious.

Spatio-temporal data have become an important aspect in the real world application. For example, the real world trajectory of an object or a person is tracked that includes geographic information collected using the geographic information systems along with the temporal characteristics. It is of prime importance to explore the interaction of both spatial and temporal aspect of the data to better understand the behavior of trajectories and identify the trend.

Visualization is an effective tool which help the users to understand the data and develop concrete ideas. Generally spatio-temporal data would be large in size and hence need an effective mechanism to represent and visualize the data by avoiding the common pitfalls such as visual cluttering and occlusion. Due to the availability of more than one attribute for visualization, it is very tempting to employ three dimensional technique to process spatio-temporal data. However, three dimensional technique has its own disadvantage like occlusion and perspective shortening. Hence in this thesis, two dimensional technique is used to solve the problem.

There has been lot of research on the spatial data and temporal data individually. But the interaction between the two data have not been studied extensively. The visualization of the combined interaction between the spatial and temporal data is of high importance to uncover the insights and identify the trends in the data.

1.2 Color As Time Dimension To Represent Spatio-Temporal Data in 2D

In this thesis, color is used as the time dimension to represent the temporal data in two dimension. Every data point has the time information that is converted into the appropriate color based on the HSV color model. The route connecting the spatial data is given color based on their time information. Due to the changes in the time, the color has a smooth transition from one color to another, that provides an interpolation of the color. Hence this approach helps the user to easily understand the data and identify the trends and gain insights.

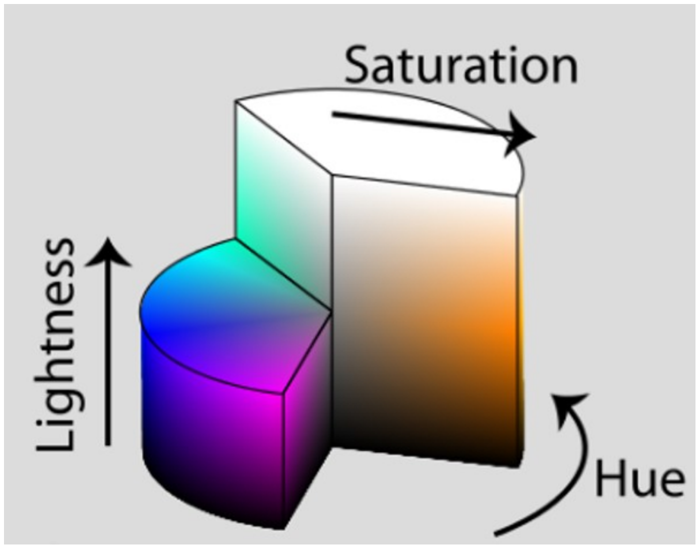


Fig. 1.1. HSV color model [14]

1.3 2D Representation Of Spatio-Temporal Data

The spatio-temporal data is visualized by representing both spatial and time information. Maps have been a effective way to represent the spatial information. Some of the common ways to represent time are third dimension and animation. However three dimension technique leads to occlusion and perspective shortening where it suffers from justifying the measures of objects in 3D due to prospective projections [13].

Additionally employing techniques based on animation to show variation of time have lacked in the exploration of the details present in the data that leads to difficulty in analyzing the trend and pattern.

In this paper[13] an empirical discussion is made on two dimension and three dimension representations of spatio-temporal data. The two dimension density map and the three dimension density cube were compared. According to the discussion [13], two dimension density map led to more accurate insights for the task related to trend detection and pattern identification. two dimension technique performs better than three dimension in case of a large cluttered data. Additionally two dimension technique proves to be more effective as it allows less time to complete the task and produce more accurate results. Comparison of the data is easier to make in two dimension and explored further in this thesis. Although three dimension techniques have its own advantage such as providing holistic views of data and reviling subtly changing patterns over time, two dimensional technique is easier to explore and user friendly.

1.4 Overview Of Visualization Technique

With ever changing growth in computer science world, there are a lot of dataset that describes spatial and temporal information used in many application fields. In this project, a novel visualization technique is proposed in the two dimensional space to represent the spatio-temporal data. The spatial data is represented using the map to locate the data points using the latitude and longitude of that data. The time specifics are represented by the HSV color model based on the time traveled. The different data points that holds range of time instances are converted to the corresponding color based on the color model. As the next step, the data points are interpolated to get a smooth curve. The interpolated points obtained finally provide knowledge on both space and time related information.

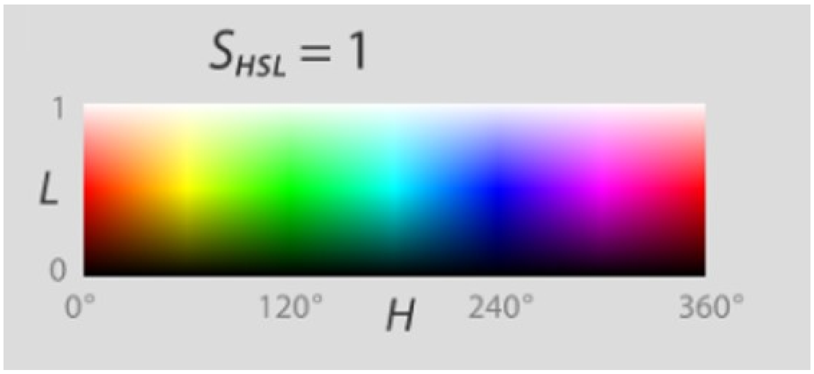


Fig. 1.2. HSV color interpolation [14]

The novel technique proposed in this thesis allows the user to identify the patterns and gain insights from the data. In addition, the visualization technique include user interaction option to help the user interact with the data and search for the specific patterns that interest them. The insights gained by the user can be useful for a range of domains such as taxi routes, post office delivery routes, airplane movement tracking, general everyday movement tracking for a person and similar applications.

1.5 Thesis Organization

Chapter 1 describes the overview of the thesis with the introduction of the spatio temporal data and describes the representation of the temporal data by color. It discusses about the benefits of using visualization in two dimensional space and the effectiveness of the proposed technique. Chapter 2 presents a literature review of the different spatio temporal techniques in the past. Chapter 3 describes the representation of the spatial data using the leaflet map and the addition of the temporal data to it using the color interpolation. It also describes about the color model used to represent the time and how the interpolation of color helps the user to identify the patterns and trends from the data. This technique also discussed about the color legend that help the user identify the colors displayed in the output visualization.

Chapter 4 introduces the taxi dataset in Beijing, China to experiment the visualization technique discussed in the chapter 3. Additionally algorithms to remove the outliers and simplification of the dataset to increase the performance of the visualization are discussed in this chapter.The experiments are conducted based on the Beijing taxi dataset and the results are discussed in this chapter. The results obtained are compared with the theoretical and experimental analysis. Chapter 6 presents conclusion and recommendations for the future work.

2. PREVIOUS RESEARCH

The review of the previously conducted research helps in formulating the novel technique proposed in this thesis. Due to the wide popularity gained by the spatio-temporal data, lot of research have been experimented in the recent past. Research in this field have been classified into three categories based on the type of the technique and results.

The integrated two dimension representation discussed in section 2.1 explains the spatial data and temporal data visualized in an integrated view resulting in two dimensional space. The link view two dimension representation discussed in section 2.2 explains the different types of link methodology created to associate the spatial and temporal visualization. The integrated 3D representation in section 2.3 defines the technique in three dimensional space where the temporal data is represented in the third axis. The Fig. 2.1 graphically explains the broad categories discussed in this chapter.

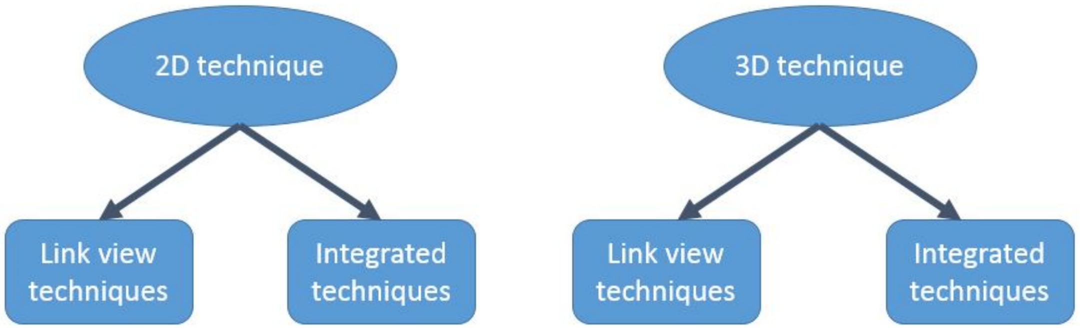


Fig. 2.1. The various categories in the previous work

2.1 Integrated 2D Representation

In this paper [3] a new dynamic visualization technique has been proposed on extending the existing system that contained spatial representation by the addition of the time dimension. It is a change based model that displays variation in the shape and size of the geographic areas within the map along with the dynamic movement process. In this animation based technique, the size of an area is observed to show changes according to the attribute value for the specific time interval chosen interactively. The technique discussed in this paper [3] helps the user to analyze the change patterns and behavior of an object over different time. Despite the animation, this method is extremely difficult to look for specific insights and overall pattern for entire data in one screen to compare against different geographic location and time.

This paper [4] proposes a new visualization technique to represent the spatio-temporal data using map. The spatial information is represented by Google map and the temporal information is represented using a graphic display that is produced within the area of it's specific geographic location as shown in Fig. 2.2. The time interval is chosen interactively by the user and the time-graph displays within every geographic location. The horizontal axis represents the time interval and the vertical axis represents the variation of selected attribute over time interval. This technique discussed in [4] to represent time graph within the area of a map is not feasible for data that contain a large time interval. In such cases, users need to choose subset of time interval to view the visualization. Such process makes it difficult to view the overall trends and patterns from the entire data.

This paper[5] proposes a two dimensional visualization technique named story-graph where parallel axes are used to represent the spatial information. The latitude and longitude extracted from the location information are represented in each parallel axis and time is represented in the horizontal axis as shown in Fig. 2.3. The change in color, size and shape of markers are used to highlight additional attributes. This technique[5] evaluates the movement of both space and time. Representation of large

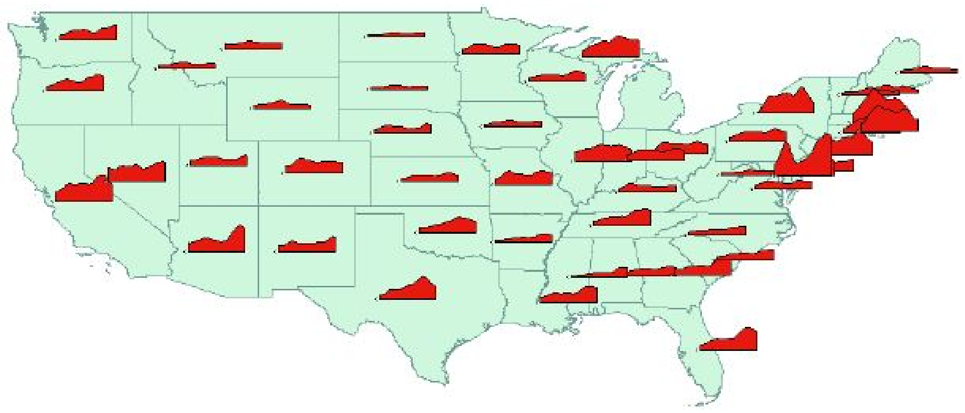


Fig. 2.2. Temporal data displayed as graph within it's geographic are [4]

scale data becomes difficult as it leads to visual cluttering and occlusion. The users are allowed to choose the specific location or time intervals interactively to reduce the visual cluttering. However it suffers from difficult depth precision as more data points are plotted. In addition to that, it is also difficult to visually perceive the spatial information as the geographic latitude and longitude are represented and not the actual location.

2.2 Link View 2D Representation

Landesberger and Bremm [6] introduced dynamic categorical data view (DCDV) technique to visualize the categorical change in space and time data. This method [6] combines the geographic map view on data locations with the DCDV as shown in Fig. 2.4. For the detailed inspection of the map view, DCDV provides a smooth transition of categories between time step. The other attributes such as width of the time bar indicate the category count during the time step and color to represent the different categories. The user has the ability to select the time step in the event of detailed inspection. The time selection is also allowed to be chosen automatically

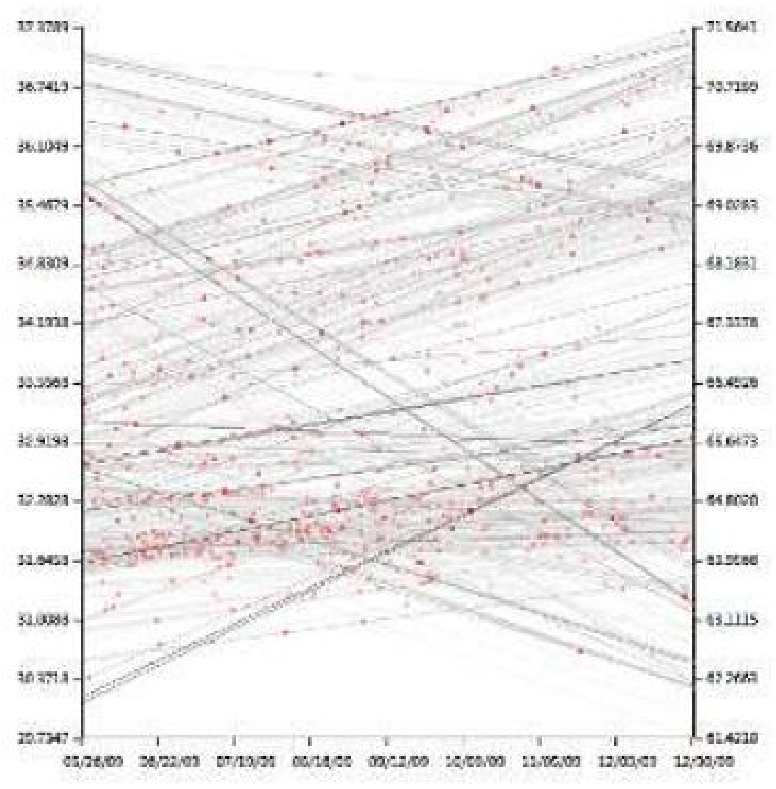


Fig. 2.3. Visualization using storygraph technique [5]

based on the algorithm that is available for global data as well as for subset of time interval. The selection of appropriate parameters for the algorithm is essential to help the user visualize the data in the right way.

One of the main concern of using this technique [6] is the scalability with regard to the number of categorical states, objects and time steps as well as the ordering of categories in view. This technique [6] is ideal for small number of categorical values whereas in the event of greater than ten categories, it leads to the over-plotting problem.

Edsall and Peuquet [7] focus on representation of time in a GIS (Geographic information system) user interface. This technique [7] utilizes multiple screens to visualize the overall data. The first screen is used for representing the spatial information,

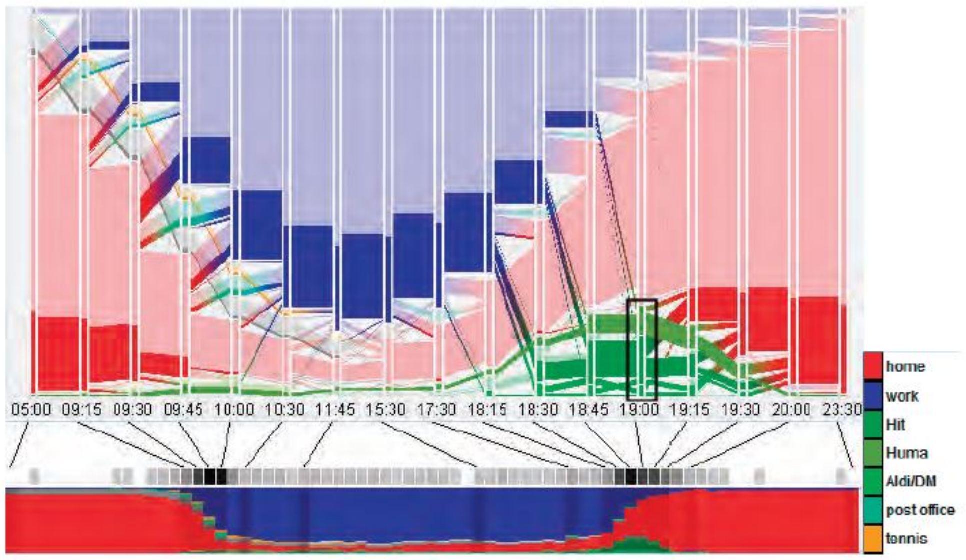


Fig. 2.4. Dynamic categorical data view with highlighting (DCDV) [6]

the second screen is occupied by the object that represents the spatial and temporal information, and final screen represents the temporal component of the data.

The technique discussed in [7] focus on the integrative approach, where the changes from the map screen interactively change the information displayed on the other screens. This technique [7] of visualization delivers a conceptual distinctions between linear and cyclical form of data. However such distinctions requires creative input that is challenging and demanding for the user to visualize the final output. It requires every user to have domain knowledge to operate on the input data.

Andrienko and Gatalsky [8] created a classification of analytic task, estimated based on the different aspect of a spatial phenomenon to vary with time. It constitutes a time manager connected to a dynamic map display. The user can interactively select the time intervals for the data presented in the map referred using the time manager. In addition, the user have the control over the data displayed on the map. It also makes the data available on the map to be represented for any specific instant of time or an interval of time period. Additionally, the overall view of the data is also displayed in the output.

However, the main drawback in this classification [8] is that it does not allow the comparison between the data at different instant of time. This have an effect on the final solution from the users to arrive at the result. This technique provide different snapshot information at various instance of time but could not help the users to formulate any overall trend or patterns using the snapshots.

2.3 Integrated 3D Representation

In this paper[9], the spatio-temporal data is visualized using a three dimensional integrated approach. Map is used to represent the spatial information. To visualize the time attribute, a new dimension is added to create the output in three dimensional space. Two different icons mentioned in [9] to represent the temporal information are pencil shaped icons and helix shaped icons. The third axis for time is included where the perception of geographic information is lost and makes the output visualization difficult to recognize the trends. The overall information is not obtained in just a single view as the user needs to rotate the map and the icons to get to the other views.

The accurate geographic information for any time data is not possible to identify accurately because the centroid of its respective areas are located with icons which are aligned with z axis of three dimensional perspective. An additional algorithm is used to position the icons to avoid occlusion and conflicts in an iterative fashion. In case of large data, the iterative global algorithm becomes very complex and possibility of ending in infinite loop. This technique allows the user to select events of interest which limits the amount of data displayed on the map. It adds restriction to the users to have domain knowledge to specify the predicts.

Hewagamage and Hirakawa in [10] discuss the spatio-temporal data displayed using a geometric shape that help in revealing the periodical patterns. It is a three dimensional visualization with spatial information displayed on the two dimensional map, and time displayed as the third dimension.

In this technique [10] time is represented using spiral or timeline based on the input type of data such as periodic data or linear data. The representation of the third dimension makes the visualization more difficult to associate the spatial information to the higher altitude of time dimension.

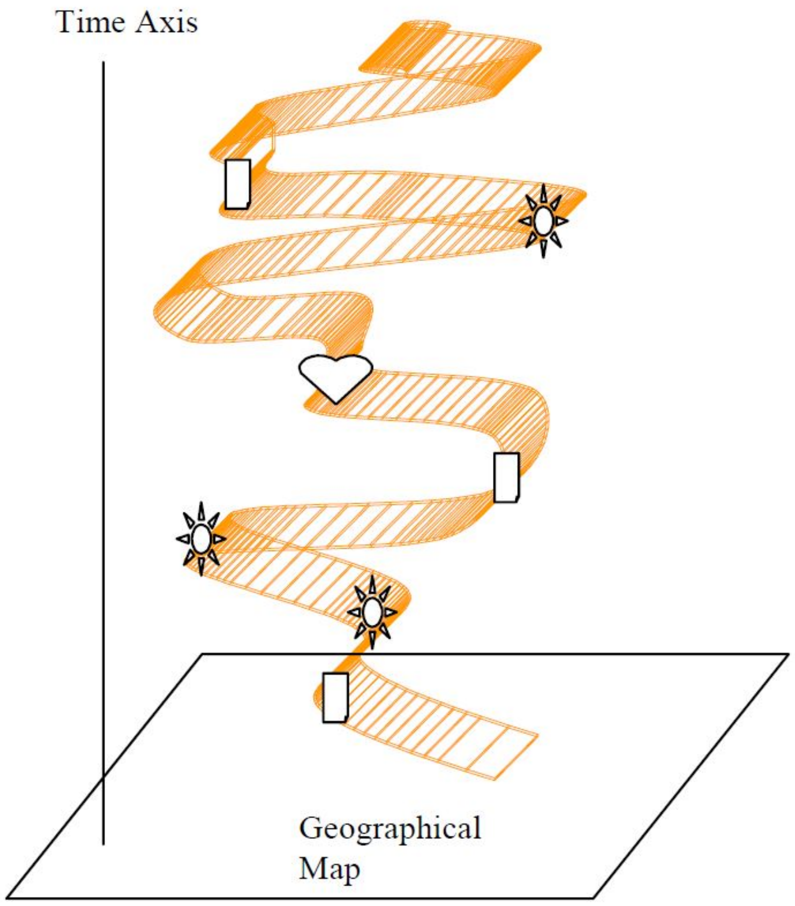


Fig. 2.5. Visualization using Spiral technique in 3D [10]

This technique [10] provides more difficulty to keep track of time in the event of high clusters, visual cluttering and occlusion. Besides the three dimension, it displays the output as shown in Fig. 2.5 that contains both space and time at the same screen which helps user to get an overall understanding of the data. When there is a vast geographical location available to visualize, the spirals or timeline will produce more visual clutter. In addition, the different icons discussed earlier represents the various attributes as chosen interactively by the user.

Andrienko [11] introduces a new technique called space time cube for the data that contains various events described on spatial and temporal references. This paper focus on the task of detecting spatio-temporal patterns in such event occurrences. The horizontal axis of the base of the cube represent the spatial information while the vertical axis in the represent the temporal dimension.

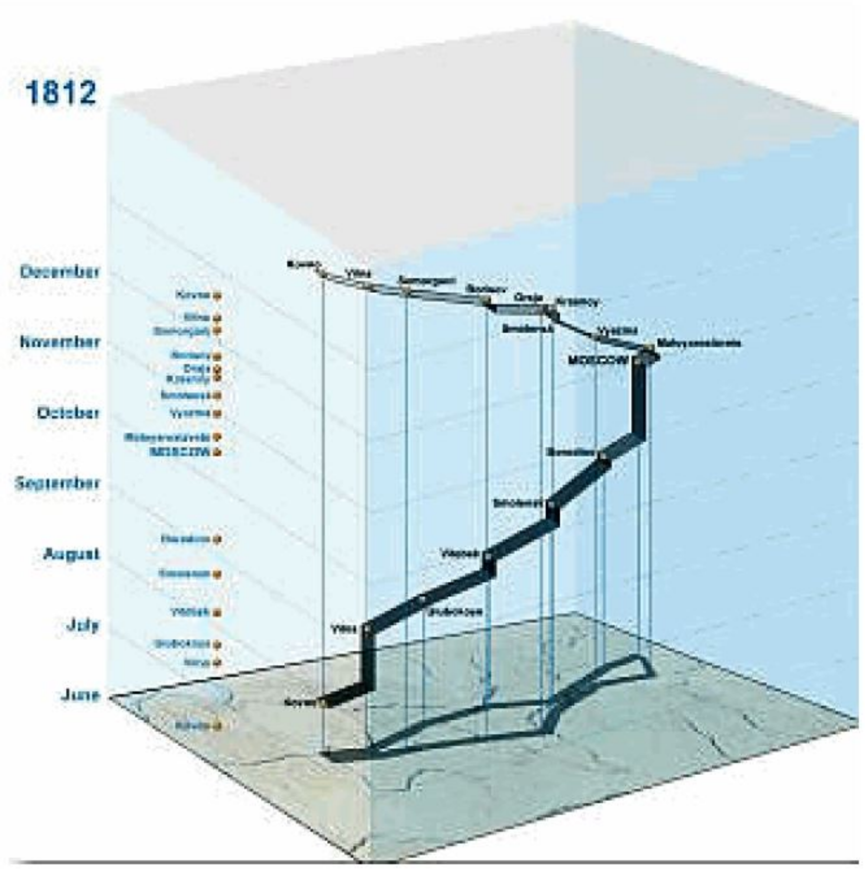


Fig. 2.6. Representation of Space time cube [11]

As events in [11] are described as circle, the color and shape of such circles are used to indicate the thematic properties of events. Fig. 2.6 depicts the three dimensional representation of space time cube with various events. The events occurred in the earlier time are represented at the bottom of cube and the events goes to the higher altitude of the cube as the time increases. In the case of large data, the third axis goes higher in the vertical axis resulting in the perspective loss of spatial information. When data aggregation is chosen for a specific time interval, it becomes more difficult to gain overall trend and pattern from the data. Moreover, Symbol overlapping also leads to visual cluttering and occlusion.

3. REPRESENTATION OF SPATIAL AND TEMPORAL INFORMATION

3.1 Representation Of Map

There are number of methods to represent the geographic spatial data such as latitude and longitude. In this thesis, map which is one way to visualize the geographic information is used. Map relate the geographic information with the real world and help us to analyze the data very efficiently. It helps the user to understand the patterns in the data by plotting the geographic points on map. Map facilitates the real features of the world to be represented like the streets, mountains, water bodies and other earth features. The Fig. 3.1 represents the map created using the JavaScript library called as leaflet.

The representation of the spatial information is implemented using JavaScript library for representing the map named leaflet. Leaflet represents the interactive maps written in JavaScript which is an efficient and mobile friendly library. It works well across all the desktop and mobile platforms. In this work, mapbox.streets are overlayed on leaflet map inside the division element in the HTML page. Additional option for the map is initialized by providing inputs for the center location of the page in the form of geographical coordinates of latitude and longitude.

There are other controls given to the map to make it more interactive such as zoom levels and attribution controls. The initial zoom level will provide the initial display for the user. The zoom level can be controlled after the initial output by zoom control user interaction option i.e. positive and negative symbol on the top left corner of the page display. The maximum zoom level can be set in the program during the creation of the map. The tile layer is added to the map by choosing the required type based on the dataset to be visualized. In this thesis, the street view tile on the map as we intend to visualize the trajectory of the taxi in the experiment.

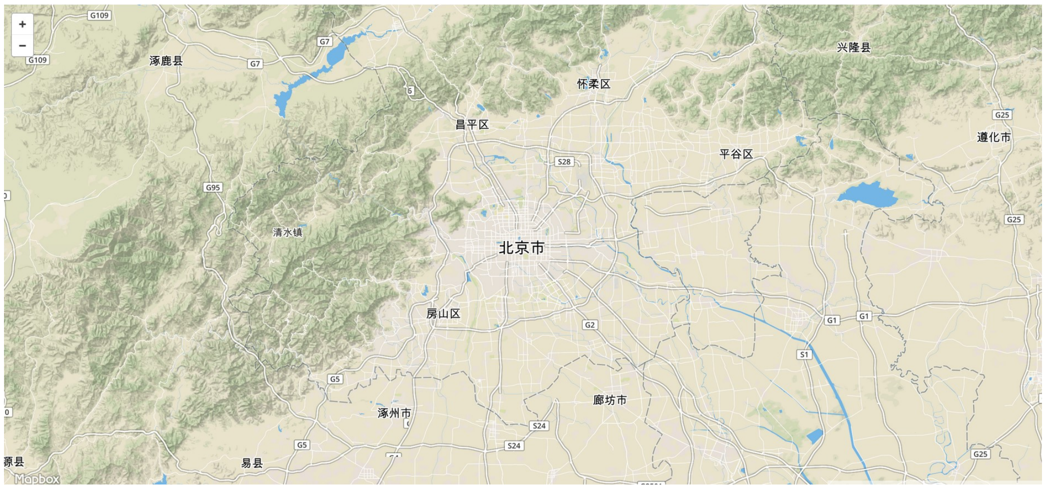


Fig. 3.1. Leaflet Map representation

3.2 Representation Of Data On Leaflet Using GeoJson

The format to represent data on leaflet is GeoJson format as it is very efficient, light and straightforward. GeoJson format specializes in storing the geographical data. It is very easy to parse to the program and a native of the JavaScript library. Leaflet supports GeoJson to represent the geographic points as they best describe the data using multiple properties. In this thesis, GeoJson format is used to represent the data using two very popular geographic data structure namely Features and FeatureCollections. The features are then interpolated using the Linestring geometry types supported by GeoJson. The GeoJson points are presented as the coordinates for the inputs to the map. Finally, the GeoJson objects are added to the map using the GeoJson layer - addMap() method.

3.3 Assigning Color To Time Dimension

In this paper, we have chosen to represent color as time dimension. Representing color as time is the novel method as shown in Fig. 3.2. In the application field such as taxi dataset that contains location of the taxi collected at various time range is a good example for applying this technique. Every row of the data contains information about the taxi location and associate time stamp. The knowledge of the location of an object along with the time stamp will help us to understand the data better. It also will reveal interesting patterns and trends. An algorithm is created to assign a color for every time stamp based on the HSV color model.

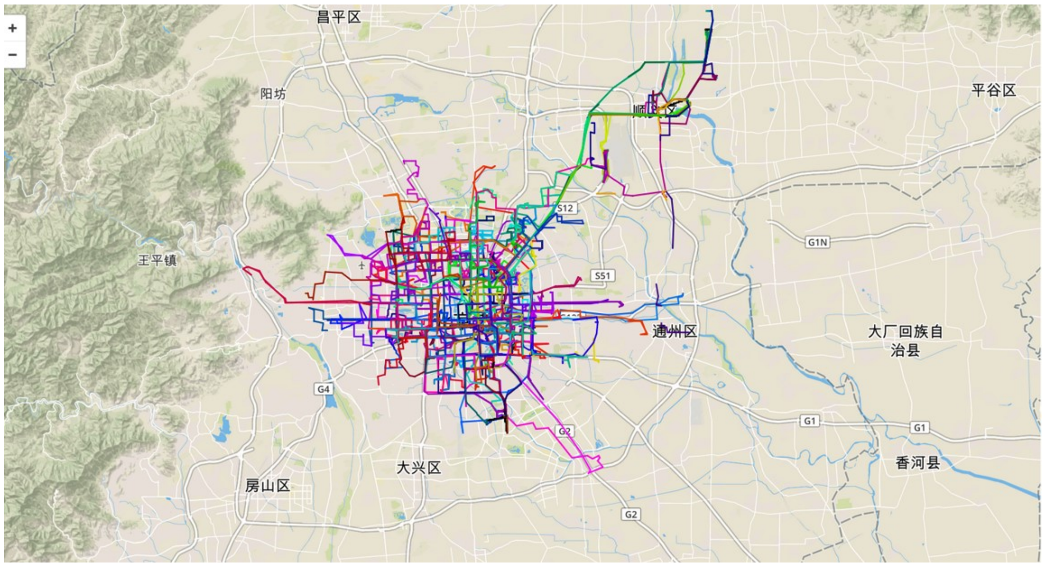


Fig. 3.2. Output visualization after assigning color to time dimension

3.4 HSV Color Introduction

HSV[14] is known for Hue, Saturation and Value. HSV color model is the most common cylindrical coordinate representation of colors in RGB color model. Hue represents the angle around the central vertical axis of the cylinder that takes the value between 0 and 360 degrees. Hue select the specific color for the representation. For example red, blue, green and the other colors have a unique value for hue. Saturation is the angle around the central vertical axis having the range between 0 and 1. High saturation of a color looks more brighter when compared to the low saturation of the same color. Low saturation of a color is dull and grayish. Value represents the distance around of the axis that represents different shades of a color ranging from black to white.

In HSV color representation, hue ranges from 0 to 360. 0 indicates the red color which changes to green color at 120 degrees, blue at 240 and back to red at 360 degrees. The value component range from 0 to 1 where 0 is black and 1 is equivalent to white. Saturation component of HSV range from 0 to 1 where 1 is bright and rich in color and 0 looks more dull and grayish.

HSV color component is the easier representation of color than RGB color space for this thesis. The three components of HSV - Hue, Saturation and Value are able to recognize individual color better than the three components of RGB Red, blue and green. The inputs are easy to provide for HSV color model compared to the RGB color model. The value taken by hue, saturation and value component of HSV allows us to select the color in a natural way when compared to red, blue and green component.

3.5 Inputs To HSV

For every data points, the H, S and V component value are determined based on the time information associated with each data point. The hue component decides the particular color based on the input it takes between 0 and 360. In order to visualize time, every individual seconds of time need to be provided with a unique color. The data contains the date and time attributes that will be used to determine the color. The time information that contains hour, minute and second part is split into individual component. Finally, total second value is calculated by converting the time to seconds in (3.1) and (3.2).

$$split\_time = Time.split(" : ") \tag{3.1}$$

$$Seconds = (split\_time[0] * 60 * 60) + (split\_time[1] * 60) + (split\_time[2]) \tag{3.2}$$

The brightness of the data path helps us to determine how time dimension has changed over different time periods. Changing the value in HSV will help us to determine the trend in different days. In this work, the change in brightness is associated to the different time period of the data. The time period for any dataset will be dynamically decided based on the data or chosen by the user. For example, if the dataset contains data for five days and time period is chosen as single day, then the value change will be reflected on each of the five days. Hence there would be more variation in the opacity when compared to the data containing 30 days of data.

3.6 Conversion Of HSV Color To RGB

The inputs for the color is provided using the hue, saturation and value component of HSV color representation. To represent the color on the map, the HSV color is converted to RGB (Red, Green, Blue) color format. The first step is to find the chroma obtained by multiplying the saturation of the color and the maximum value for which the chroma is maximum. Let hue, saturation and value be the components of the HSV color model. The CHROMA is calculated by the equation (3.3)

$$CHROMA = VALUE * SATURATION_{HSV} \tag{3.3}$$

The next step is to find the R1 , G1 , B1 based on the values of HUE_NEW which is obtained by the division of hue H by 60 degree.

$$HUE\_NEW = HUE/60^{\circ} \tag{3.4}$$

The value of R1, G1 and B1 is calculated based on the value of HUE_NEW component derived using the formula (3.1). If the value of HUE_NEW is undefined, then each component of R, G, B takes the value of 0. Based on the interval of

HUE_NEW between 0 and 6, the components takes CHROMA, TEMP_VALUE or ZERO. TEMP_VALUE is obtained by the equation specified in (3.5).

$$TEMP\_VALUE = CHROMA * (1 - |HUE\_NEW \bmod 2 - 1|) \qquad (3.5)$$

The value of M in (3.6) is derived by finding the difference of value of HSV component and chroma deduced in (3.1) for the specific color.

$$M = VALUE - CHROMA \qquad (3.6)$$

Based on the value of the R1, G1 and B1 values calculated earlier and the value of M derived from equation (3.6) , the value of R, G and B are estimated in (3.7).

$$(R, G, B) = (R1 + M, G1 + M, B1 + M) \qquad (3.7)$$

Finally, the RGB value is converted to the hexadecimal format to give the input for the map.

3.7 Dynamic Decision of Time Period Based On Input Data

The representation of time as color depends on the necessity to decide the time period that constitutes one cycle. After one cycle, the color will repeat itself with the different saturation and value. The decision of one time period is decided based on the data, as the time duration can contain data points anywhere between one hour to multiple year. Hence, the cycle duration is determined based on the length of total time duration to visualize. In our case, the data contains traffic information for multiple taxi of a week. In this case, the cycle is repeated for every day.

This allows us to gain multiple insights by viewing the same color with value for different time period. In other scenarios like the data containing information for a day, then one time period will represent one hour, or can be split between morning, afternoon, evening and night. If the data contains an year data, then one time period could represent the different season such as summer, winter, spring and autumn or for every month. The decision between choosing the single time period depends on the data and the kind of information looking for. The decision to find the time period is given to the user. The user can provide inputs or the program will automatically decide the time period.

3.8 Color Interpolation

The time information for every data points are converted into color and interpolated along the trajectory. The inputs are converted to the HSV color space because interpolation is easier to provide inputs in HSV color space when compared to the RGB color space. Additionally, interpolation in HSV color space is closely related to the perception of the human brains and create the natural interpolation between different colors. HSV color space avoids any additional colors and reduce noise to create a smooth transition from one color to another. Finally, the HSV color is converted back to RGB color and again converted to hexadecimal value to provide the input for the interpolation.

A single interpolated line string depends on the time of extreme data points. If the time interval between the end points are greater than the threshold time interval, temporary points are dynamically created between the end points. The additional points are created in such a way that the time interval between any two data points results below the threshold limit of time. These additional points are created to produce a smooth interpolation between different colors. These points are not created when the time interval between the end points is within threshold limit because, it improves efficiency and results in minimum calculation to split the points into multiple instances.

3.9 Color Legend

Different colors are used in the visualization to represent the temporal dimension on the map. The user need to understand the corresponding time for the different colors to get a better perspective about the data. The user need to understand the date and time instantaneously when they looks into the color legend. Hence in our work, we created a color legend to represent all the colors for the time interval used in the visualization.

$$x(t) = a\cos(t) \tag{3.8}$$

$$y(t) = a\sin(t) \tag{3.9}$$

$$z(t) = b * t \tag{3.10}$$

The color legend is represented using the 3D spiral called as helix. The 3D spiral starts from a point and move furthest in the y axis away as it revolves. The number of rotation in the spiral depends on the time period that needs to be represented using the legend. The radius of the 3D spiral depends on the smoothness of the color representation required and the number of colors represented for one time period. As the spiral goes upward on the y axis, the value of the HSV color also increases. The label on the side of the spiral indicates the change in the time period for every rotation. The formula to create the spiral helix is provided in the equation (3.8), (3.9) and (3.10) where radius of the circular helix is assumed to be a and the slope is b/a.

The color legend is created using three.js, which is a JavaScript API used to create and display animated three dimensional computer graphics in the web browser. Three.js contains multiple features such as animation, light, camera, geometry and objects. It helps us to create the three dimensional graphics, add event and animation to make the geometry interactive.Helix is constructed by creating multiple three dimensional plane geometry and position them based on the formula provided in the equation (3.8), (3.9) and (3.10). Iteratively each plane is given the unique color according to the input time period and given the three dimensional position for x, y and z axis.

In the Figure Fig. 3.3 it is clearly observed to see the smooth interpolation of color along the spiral form in the y axis. The color repeat in the next spiral rotation

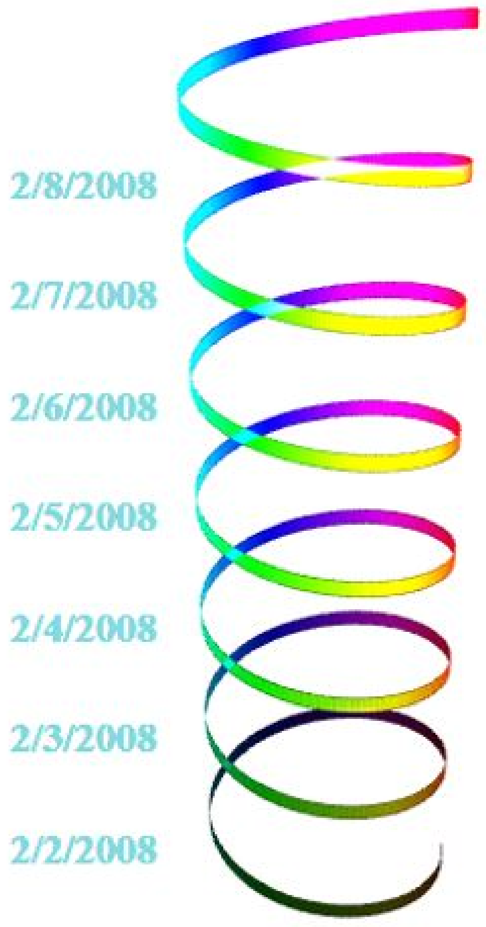


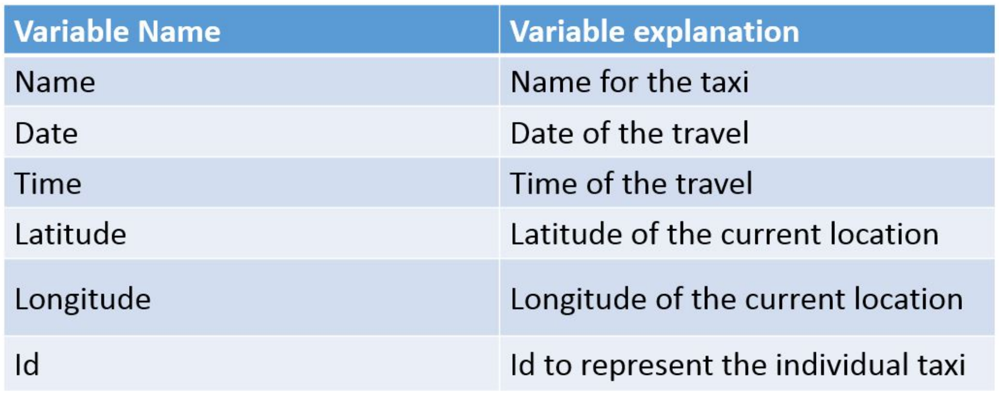
Fig. 3.3. Color Legend

with the different saturation level in the HSV scale that indicates the changes in the time period. For example the time period depend on the input dataset which can range from a day, week, hour or any other period.

4. EXPERIMENT AND RESULTS

4.1 Data Set Explanation

The experiment is conducted based on the Beijing taxi dataset [1]. T-Drive trajectory dataset that contains a one-week trajectories of 10,357 taxis. The total number of points in this dataset is about 15 million and the total distance of the trajectories reaches 9 million kilometers. The dataset contains variables like latitude and longitude to identify the location. The temporal information are observed by the date and time variables. Date is given in the format of MM/DD/YYYY where M is month, D is date and Y represents year. Time is in 24 hour format HH:MM:SS where H represents hour, M represents minute and S represents second. The Fig. 4.1 explains the different variables in the dataset along with the explanation.



Variable Name	Variable explanation
Name	Name for the taxi
Date	Date of the travel
Time	Time of the travel
Latitude	Latitude of the current location
Longitude	Longitude of the current location
Id	Id to represent the individual taxi

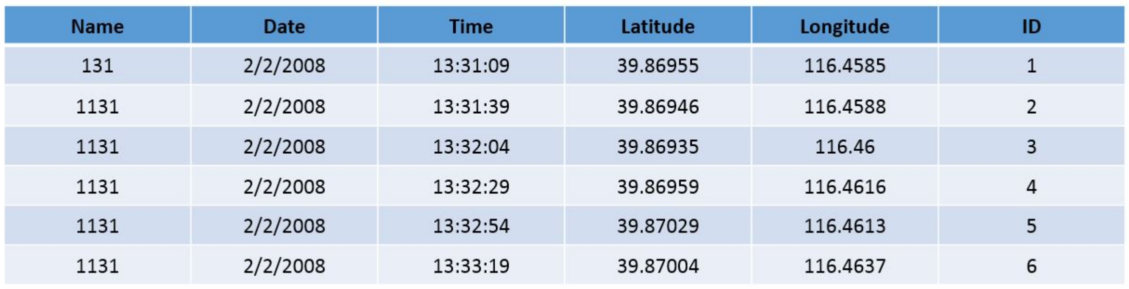
Fig. 4.1. Variable description

Every taxi contains one-week trajectories in Beijing, China and with the above mentioned variables. The data had been collected between 2/2/2008 to 2/8/2008. The dataset is in CSV (comma separated values) format that allows to import the data for the visualization as shown in Fig. 4.2. The GPS collects the data for the

Table 4.1
Taxi dataset variable explanation

Variable name	**Variable explanation**
Name	Name of the taxi
Date	Date of the taxi
Time	Time of the travel
Latitude	Latitude of the current location
Longitude	Longitude of the current location
Id	Id to represent the individual taxi data point within each taxi

current location for every 30 seconds during the time the taxi carried the passenger. If the taxi stay at the same location for a longer time, then it is considered to be idle and waiting for the passenger to arrive.



Name	Date	Time	Latitude	Longitude	ID
131	2/2/2008	13:31:09	39.86955	116.4585	1
1131	2/2/2008	13:31:39	39.86946	116.4588	2
1131	2/2/2008	13:32:04	39.86935	116.46	3
1131	2/2/2008	13:32:29	39.86959	116.4616	4
1131	2/2/2008	13:32:54	39.87029	116.4613	5
1131	2/2/2008	13:33:19	39.87004	116.4637	6

Fig. 4.2. Sample data from the Taxi dataset

4.2 Outlier Treatment

The dataset has captured data with the frequency of every second in most of the case. But there is a deviation in some instance, where there are wrong spatial information recorded by the device. In the case of Latitude and Longitude, there are numerical error with the values obtained wrongly recorded in the GPS system that makes that point as an outlier. The time recorded for the spatial information can be recorded with errors. Such outliers cause the pattern of the trajectory data to deviate from the original path. Additionally it adds visual clutter that makes the user unable to observe the patterns and insights from the data. Hence, such outliers need be removed to get an accurate visualization and to increase efficiency in the display of the data.

4.2.1 Algorithm To Remove Outlier

The outliers are removed from the dataset by implementing outlier treatment algorithm. Based on the algorithm, the distance traveled between the current and the previous data point is calculated using the latitude and longitude of both the points.

The distance between two data points is calculated given the latitude and longitude of those points as mentioned in [15]. The distance is calculated in meters using the Haversine formula. The next step is to estimate the MaxDistance (meters per second) traveled by a taxi for the same time period as the distance is calculated. If the traveled distance is greater than the estimated max-distance, then the data point is considered as an outlier and removed. The rest of the data point are retained for the further visualization. The algorithm is explained visually in Fig. 4.3.

The figure Fig. 4.5 shows the visualization of the geometric points before applying the outlier treatment algorithm. The straight line that goes straight towards the outside the screen are the outliers. The deviation of the latitude and longitude that are captured with errors reflects the outliers.

The figure Fig. 4.6 represent the visualization after applying the outlier treatment. In the course of removing the outlier data points, there is a reduction of 20 % to 30 % of the total data points which leads to higher speed in the data processing.

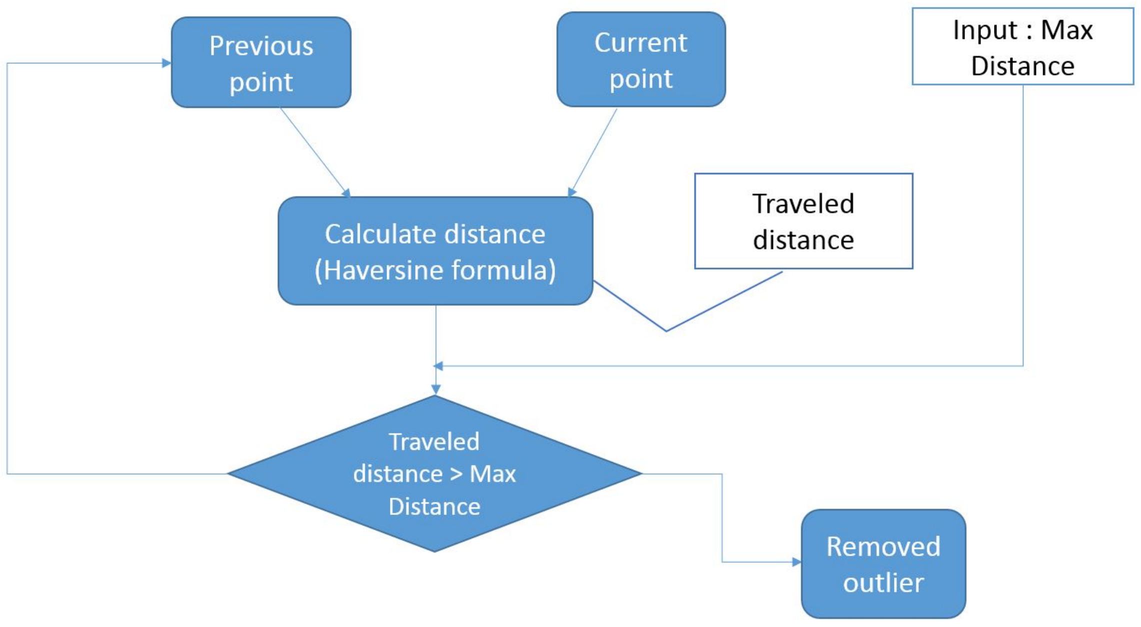
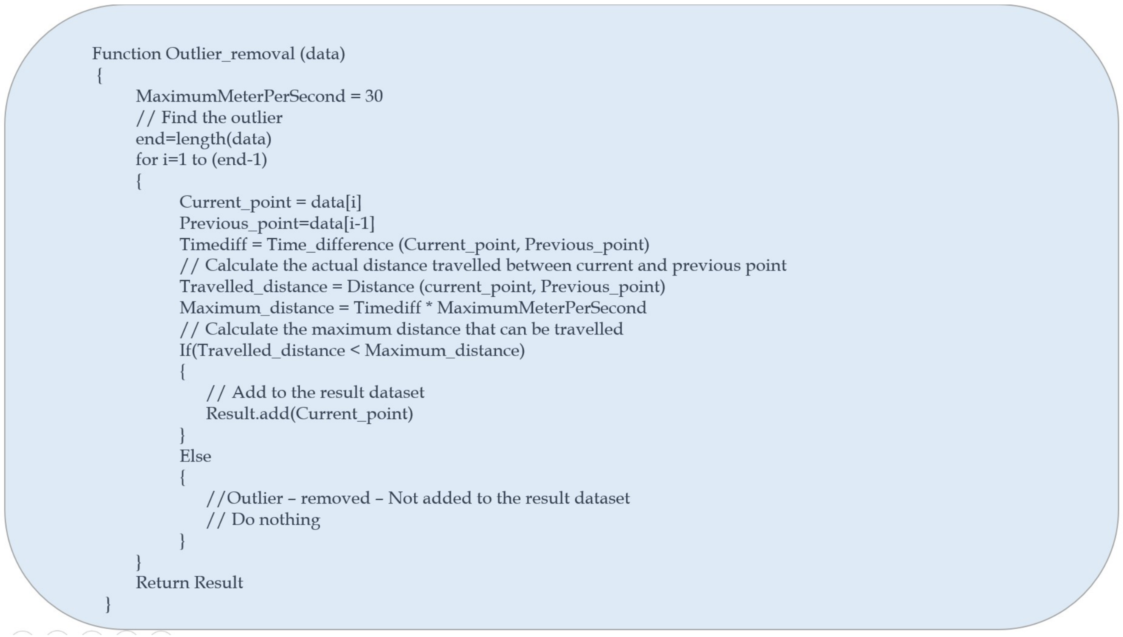


Fig. 4.3. Outlier treatment algorithm

4.2.2 Variation Of Maximum Distance Traveled

The maximum distance that can be traveled by the taxi is determined by assuming the value of the maximum MeterPerSecond metric. It is assumed that the maximum speed of the vehicle can be 70 Miles per hour which is converted to meters per second as 30 m/sec. Although this is the calculated metric, it can be varied by giving different inputs to choose the best constant value.

From the output based on different input value of maximum meters per second, we chose the optimum value. If the value for maximum meters per second is less, then we loose a lot of data points as many of the points are categorized as outliers by the algorithm. Hence we get the interpolation of fewer data points that miss the actual trajectory. But when the maximum meters per second is greater than 30, there is no difference in the output trajectory. Hence we choose the optimum value of 30 meters per second.



```
Function Outlier_removal (data)
{
    MaximumMeterPerSecond = 30
    // Find the outlier
    end=length(data)
    for i=1 to (end-1)
    {
        Current_point = data[i]
        Previous_point=data[i-1]
        Timediff = Time_difference (Current_point, Previous_point)
        // Calculate the actual distance travelled between current and previous point
        Travelled_distance = Distance (current_point, Previous_point)
        Maximum_distance = Timediff * MaximumMeterPerSecond
        // Calculate the maximum distance that can be travelled
        If(Travelled_distance < Maximum_distance)
        {
            // Add to the result dataset
            Result.add(Current_point)
        }
        Else
        {
            //Outlier – removed – Not added to the result dataset
            // Do nothing
        }
    }
    Return Result
}
```

Fig. 4.4. Outlier treatment algorithm pseudo code

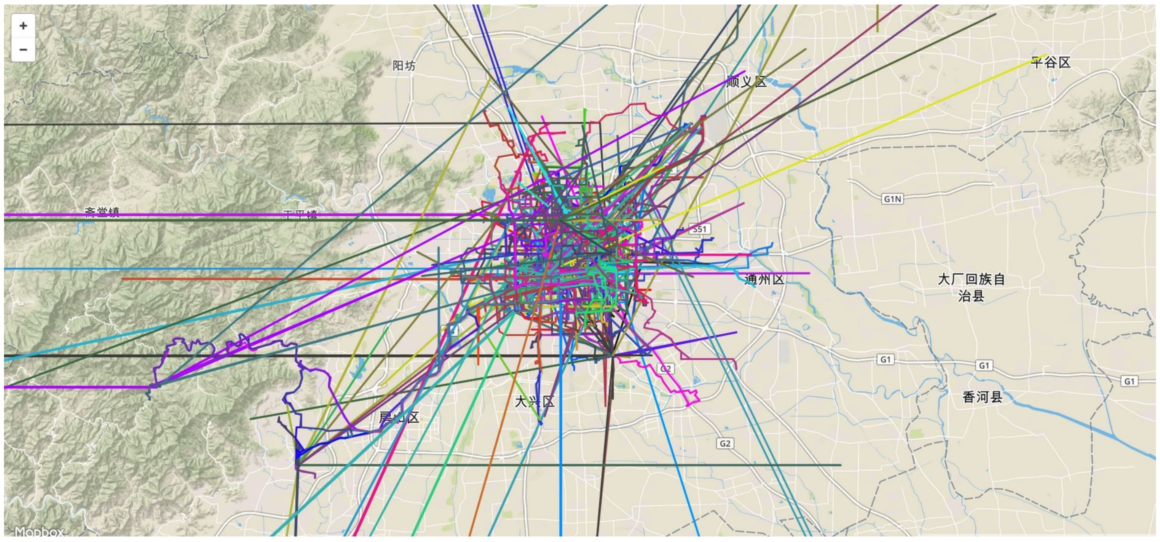


Fig. 4.5. Outlier treatment algorithm pseudo code

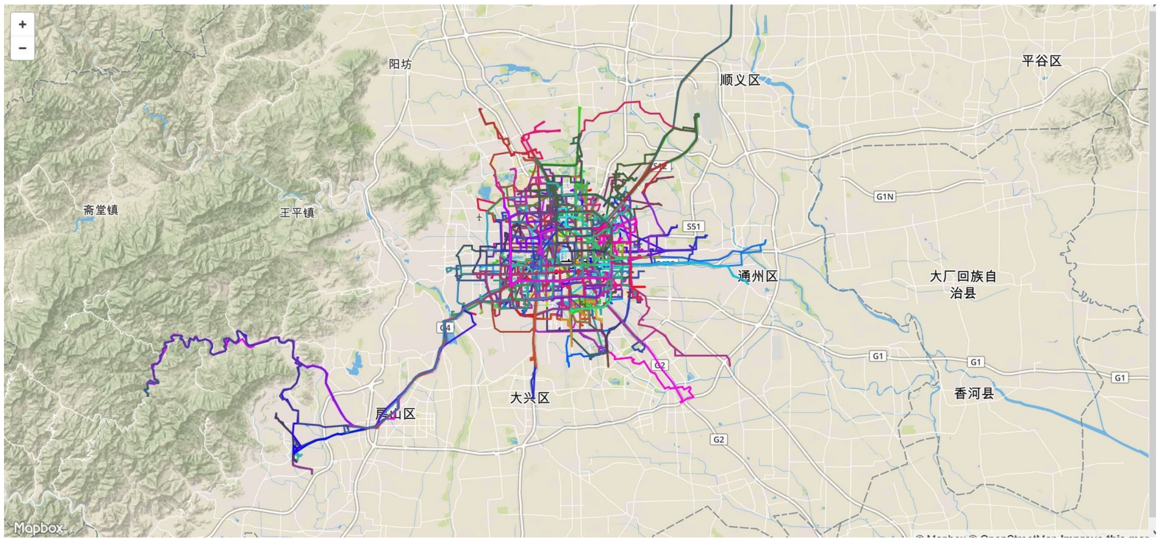


Fig. 4.6. After Outlier treatment

An experiment has been conducted by varying the value of MaxMetersPerSec input to the visualization. The Fig. 4.7 arrange the output for different inputs of MaxMetersPerSec. If the value of MaxMetersPerSec is very low when compared to the normal value, many points are determined as outliers and hence results in the distortion of the output. If the value of MaxMetersPerSec is higher than the normal value, the result would be the same and hence takes more time to load the visualization. Hence choosing the appropriate MaxMetersPerSec value is very essential for high performance and efficiency.

4.2.3 Missing Data

In the spatio temporal large datasets, missing data is one of the common problem. These datasets often contains gaps in the time recorded because of the situation under which the data measurement are acquired. There could be a number of reason that are attributed to the data missing such as the type of GPS instrument utilized to record the spatio temporal data, device malfunction, or under the influence of the

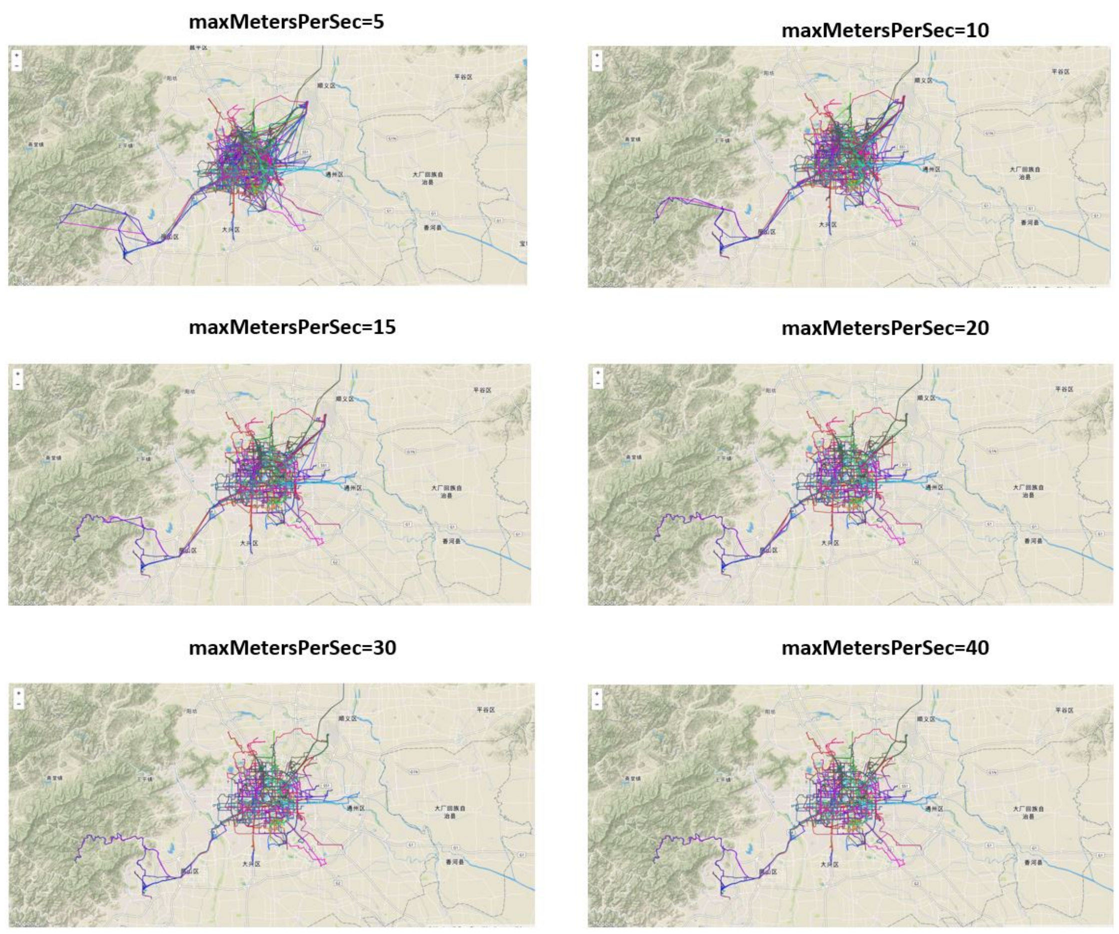


Fig. 4.7. Variation of maxMetersPerSec in Outlier treatment

taxi driver who would have manually turned off the GPS due to the in activeness or no passenger at the moment. Hence due to various reason, the missing data might will affect the visualization trend and cause trouble for the user to identify trends and gain insights. In such case, a technique has been implemented to overcome the missing data issue.

The data is identified to be missing for a short period of time and then resumed normally from a different location. Hence the interpolation of these two points would create the straight line interpolation and cross the existing paths as it does not appear to be moving on the road. Such path crossing are considered as noise and need to be

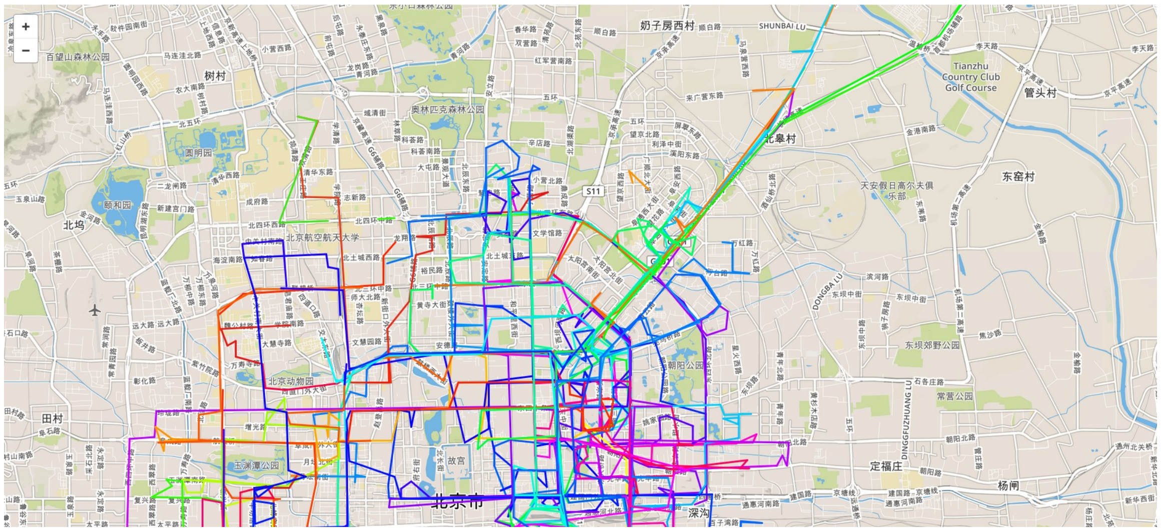


Fig. 4.8. Missing data - Threshold distance of 3000 meters

removed. In such case, the current path disconnected and a new path will be resumed from the different new location.

The first step is to identify the gap or data missing in the dataset. To identify the gap, we need to find the traveled distance and max distance that can be traveled between every successive points. The traveled distance is the distance traveled between two successive data point with successive time period. The max distance is the maximum distance that can be traveled for the given mode of transport. Max distance is the input given by the programmer based on the mode of transport such as car, bus, train and others.

For example if the mode of transport is taxi, the maximum distance that can be traveled is 31 meters per second based on the condition that the taxi can go maximum at 70 miles per hour. The threshold distance is the distance traveled in meters which is set as the limit for the data missing. If the data is missing for a certain duration and also traveled a certain distance during that missing time period, then the total distance traveled should be greater than the threshold distance to be determined as missing data that created a gap in the dataset. When the traveled

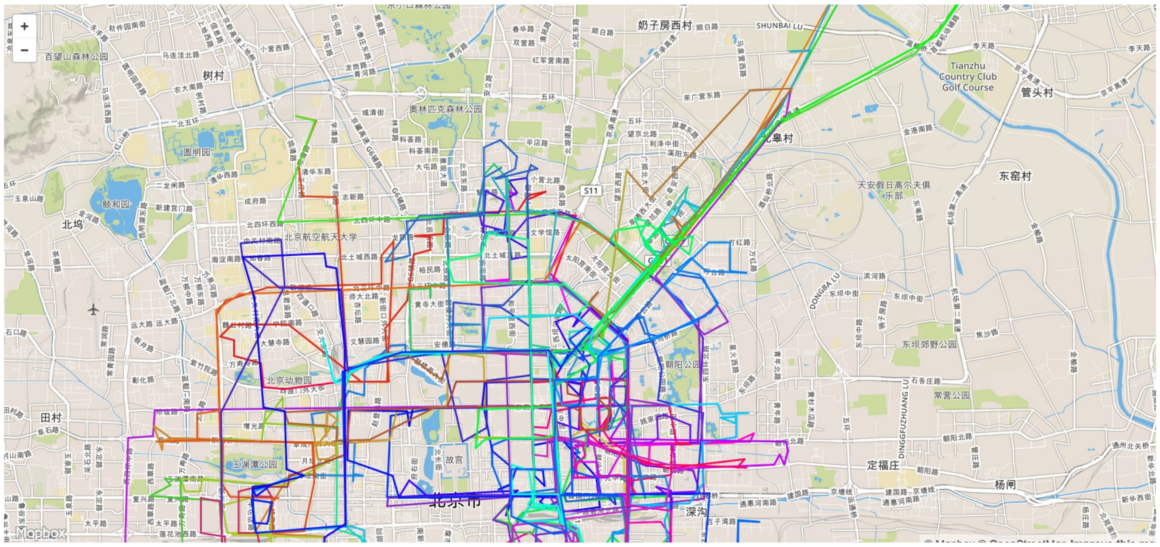


Fig. 4.9. Threshold distance of 6000 meters

distance is greater than max distance, then those points are identified as outliers. But when the traveled distance is lesser than max distance but greater than the threshold distance, then those points are categorized as data missing and hence the current path will be terminated. The new path will be created starting after the missing data is resumed. In Fig. 4.8 the threshold distance of 3000 meters is shown and the output is visualized and in Fig. 4.9 the threshold distance of 6000 meters is shown for the output visualization.

4.3 Simplification Algorithm

The main issue with the usage of the GPS trajectory data is the high volume and huge size. It has millions of rows producing data for every few seconds with the current location in latitude and longitude along with the other details. Hence using such high volume of data results in the slowing of the visualization to load.

The advantage of Leaflet maps for visualization is that it only takes time for the first time to load and after which the zoom in and zoom out option load from the current data. However, there is an initial loading time taken by the visualization due to the large amount of data. The outlier treatment algorithm have removed around 70% of the total data points from the original dataset. Despite that, there is a large dataset needs to be visualized. Hence, the number of data points need to be reduced and retain the original curve for an effective visualization, It is achieved by using the curve fitting algorithm to create the best fit of the curve.

There are many algorithm to reduce the number of points on the curve. In this thesis, the combination of simplification using Radial distance and Ramer Douglas Peuker algorithm is create to produce a simplified curve that contains the subset of points from the original curve. The combination of the two algorithm produce an efficient and accurate result as graphically represented in Fig. 4.10.

4.3.1 Radial Distance Simplification Algorithm

The first step in the algorithm is to reduce the number of data points by performing the simplification based on the radial distance method. The input to the function is the original dataset and the threshold value i.e epsilon. The square distance between every point and its previous point is calculated. If the distance is greater than the provided epsilon value, then the current point is added to the result dataset. Otherwise, the data point is removed from the output result dataset. The pseudo code in Fig. 4.11 explains the algorithm to apply the simplification using radial distance.

4.3.2 DP Simplification Algorithm

The input to the algorithm is the dataset obtained from the previous step (Radial distance simplification) with all the data points and the value of distance dimension threshold (epsilon) greater than zero. This algorithm iteratively selects all the point between start point and the end point. It automatically marks the first and last point to be kept. Based on the perpendicular distance from the end points to all the point, it finds the single furthest point. This point is the furthest point from the

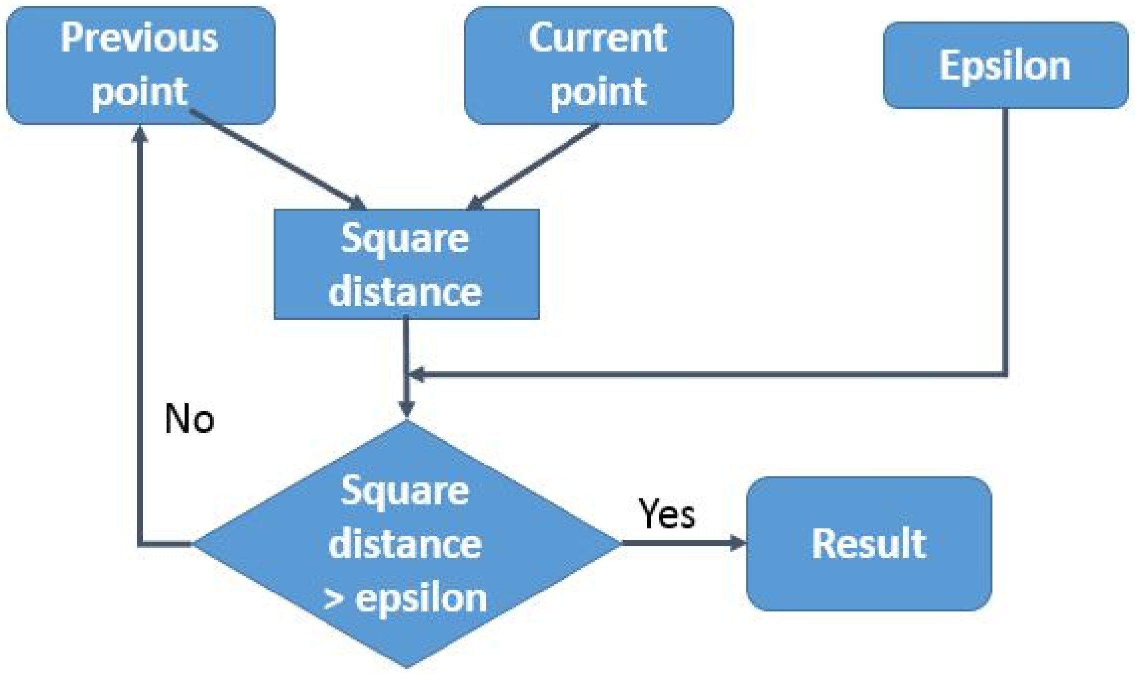


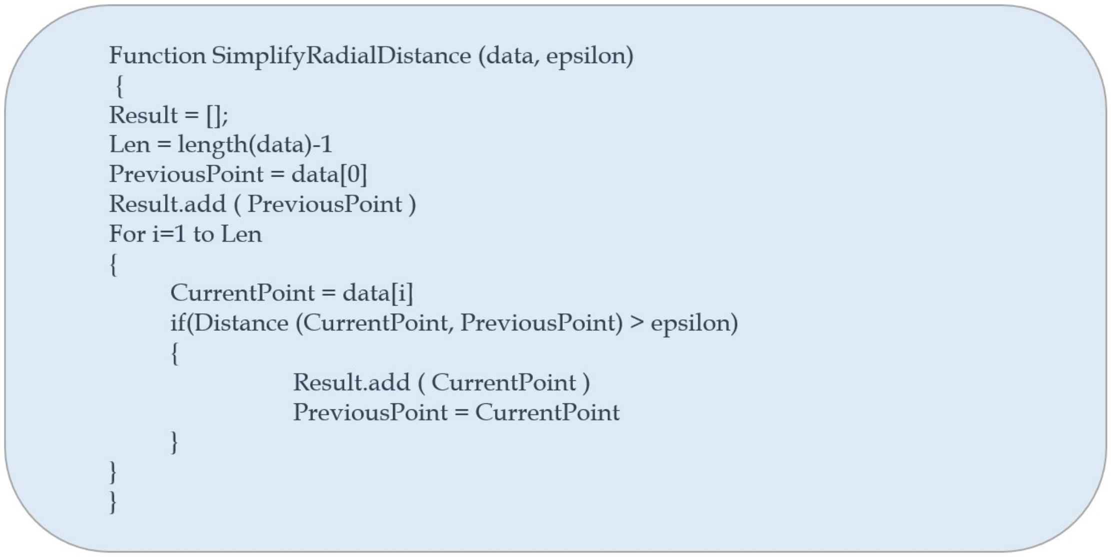
Fig. 4.10. Radial distance simplification algorithm graphical representation

line segment that connects the end points. If the distance between the point and line segment is greater than epsilon, the point is retained, else that point is discarded.

The algorithm is continued recursively by calling first the first point and the chosen point, second the chosen point and the end point. Hence, the final output is the set of marked points that form the simplified curve similar to the original curve. Pseudo code in Fig. 4.12 explains the algorithm for the dataset named data with an epsilon value and the output is stored in the result variable.

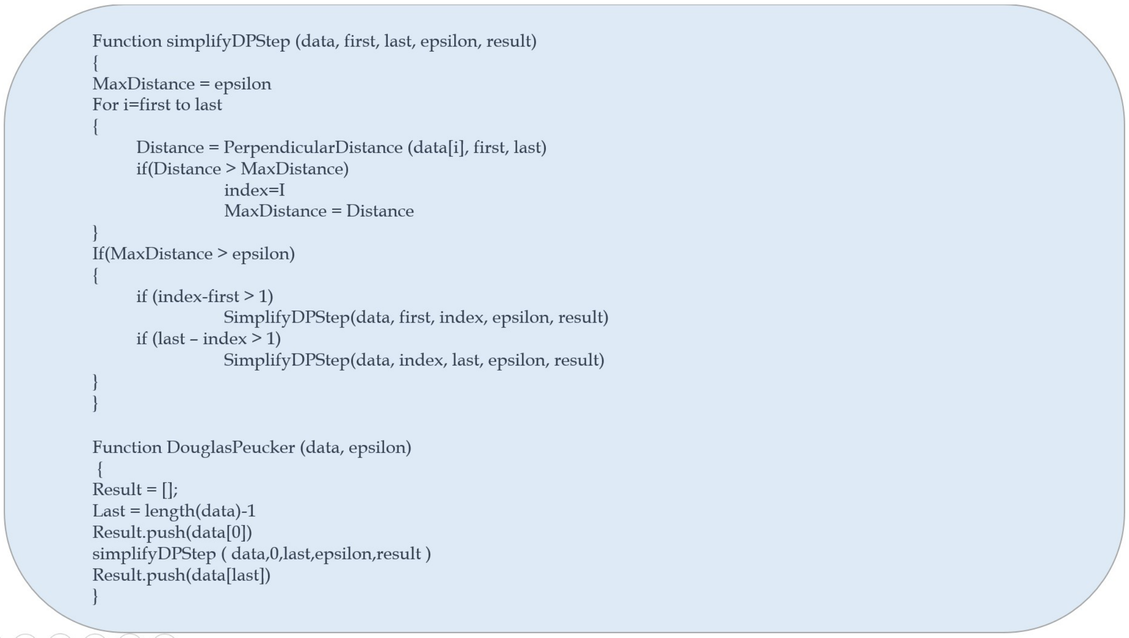
4.3.3 Overall Simplification Algorithm

The flowchart in Fig. 4.13 explains the flow of the input dataset to the final reduced size dataset. The total number of data points reduced depends on the value of the epsilon. The higher the value of the epsilon, lesser the number of data points



```
Function SimplifyRadialDistance (data, epsilon)
 {
Result = [];
Len = length(data)-1
PreviousPoint = data[0]
Result.add ( PreviousPoint )
For i=1 to Len
{
        CurrentPoint = data[i]
        if(Distance (CurrentPoint, PreviousPoint) > epsilon)
        {
                        Result.add ( CurrentPoint )
                        PreviousPoint = CurrentPoint
        }
}
}
```

Fig. 4.11. Radial distance simplification algorithm pseudo code



```
Function simplifyDPStep (data, first, last, epsilon, result)
{
MaxDistance = epsilon
For i=first to last
{
        Distance = PerpendicularDistance (data[i], first, last)
        if(Distance > MaxDistance)
                        index=I
                        MaxDistance = Distance
}
If(MaxDistance > epsilon)
{
        if (index-first > 1)
                        SimplifyDPStep(data, first, index, epsilon, result)
        if (last – index > 1)
                        SimplifyDPStep(data, index, last, epsilon, result)
}
}

Function DouglasPeucker (data, epsilon)
 {
Result = [];
Last = length(data)-1
Result.push(data[0])
simplifyDPStep ( data,0,last,epsilon,result )
Result.push(data[last])
}
```

Fig. 4.12. Simplify DP simplification algorithm pseudo code

produced in the output. After applying the above mentioned simplifying algorithm, the time taken to load the simplified curve has considerably reduced compared to the original curve.

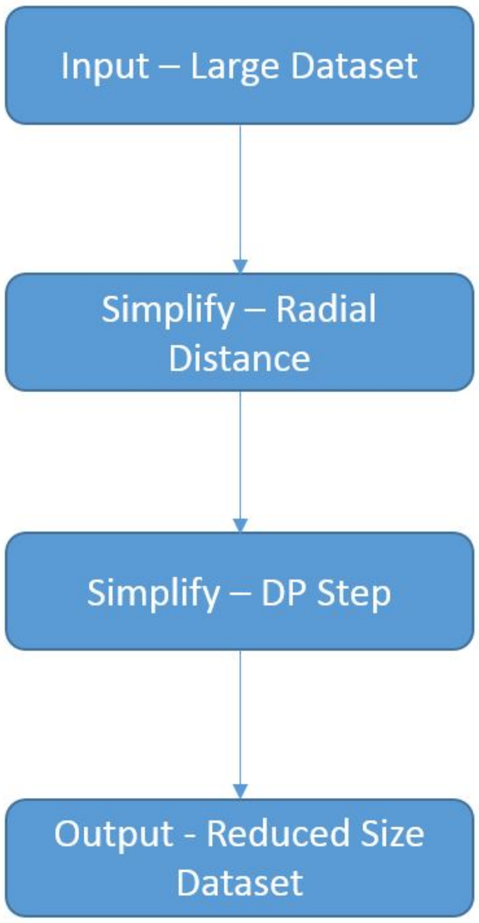


Fig. 4.13. Overall flow of simplification algorithm

4.3.4 Variation Of Epsilon Value

The input parameter to the simplification algorithm is epsilon which can be controlled by the user according to the requirements. The smallest value taken by epsilon is zero. The smaller the value (less than 0) taken by the epsilon, better the performance of the output visualization. However, the number of points in the result dataset is higher due to the smaller value of epsilon that results in more time to load the visualization.

The value of epsilon can be provided with the higher value (less than 0), the performance of the output visualization is distorted. However, in such case the time taken to load is visualization will be low as there are less number of data points in the result dataset. The variation in epsilon value is experimented using the taxi dataset and the results of visualization is compared in Fig. 4.14.

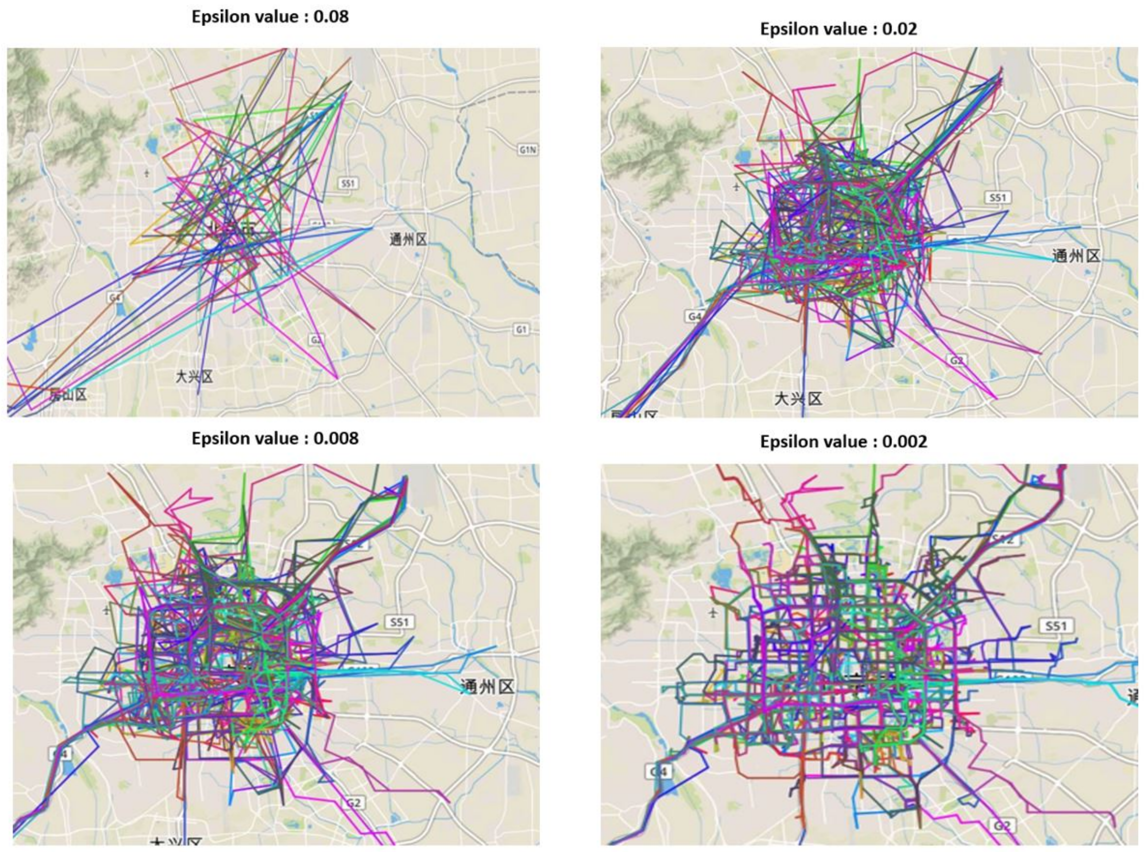


Fig. 4.14. The output visualization for different input values of epsilon

4.3.5 Performance Of The Simplification Algorithm

The performance of the visualization after applying the simplification has resulted in high performance of the overall result. With the decreased number of data points in the final result dataset, the browser takes less time to load the data for the visualization. It is also highly useful for the user interaction. As the user would like to perform the user interaction option like zoom in and zoom out, it would produce the result almost instantaneously. The performance of the simplification algorithm is evaluated for the dataset that has 376,000 rows and the results are displayed in Fig. 4.15.

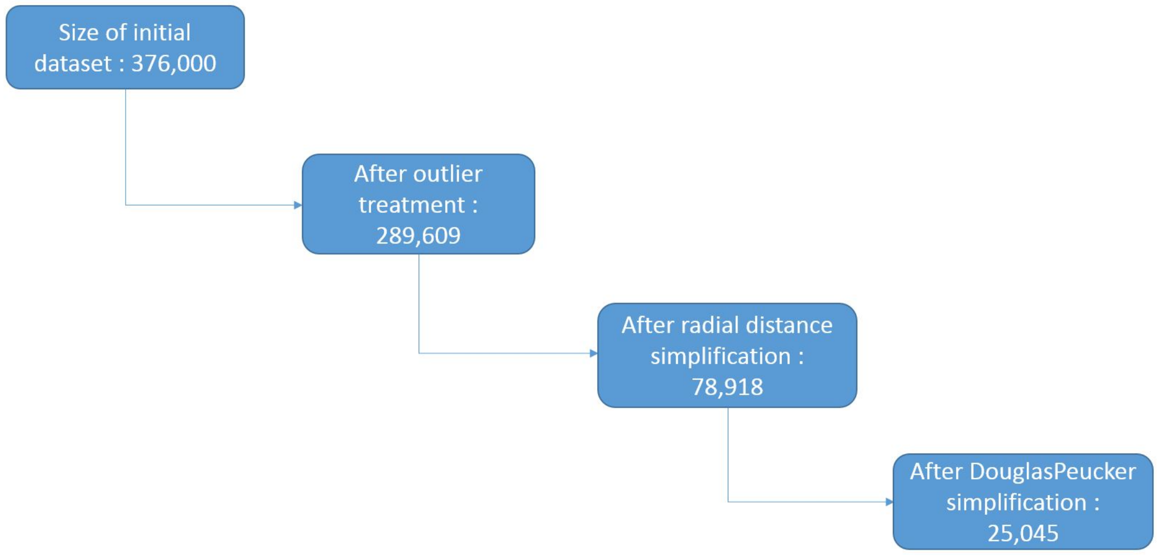

Size of initial dataset : 376,000
After outlier treatment : 289,609
After radial distance simplification : 78,918
After DouglasPeucker simplification : 25,045

Fig. 4.15. Performance of simplification algorithm

4.3.6 Large Data Overlapping Issue

Visualization of large set of data points allows the user to identify important insights and the relationship that exists between the different group of points. However, managing such large data sets lead to overlapping of the data points and visual clutter. One of the major challenge for the trajectory data that we use, is that they tend to overlap when the set of data points travel in the same location. One type of solution is that there is a natural error with the methodology of capturing the Geo-coordinates of latitude and longitude for the current location of the taxi. This natural error have lead to the distinct points for the taxis on the same location. Hence interpolation of such points have lead to distinct path.

The second type of solution to the overlapping of path in visualization relies on the user interaction available. Users can use the option of filter, zoom in, to make the visualization available for just the data requested by the user. Zooming option helps the user to zoom to the required location and can clearly view the difference between the different path. The advantage of using the zoom option with the leaflet map is that, leaflet does not take additional time to load the data point, which produces the output almost instantaneously. Hence the user can easily zoom in and zoom out to the required part of the map with very less overhead and helps them to identify insights.

The usage of the Ramer Douglas Peuker algorithm to reduce the number of data points to produce the simplified curve has proven to be advantage for the solution for data overlapping issue. According to the algorithm, it produces a subset of data points which is used in the visualization of the trajectories of the taxi. Additionally we know that there is a natural error that occurs while capturing the geographic location. Hence, the interpolation of the reduced dataset leads to the separate distinct path. Since there is a small variation of latitude and longitude for every location, interpolating these points create a separate line segment for all the distinct path. These distinct path is clearly visible when the user is able to zoom in to a certain location. The visualization of such distinct path helps the user to clearly identify the insights and conclude important relationships. When the different path are traveled in different time period, the user can analyze the distinct path with the perspective of the location and time information and conclude complex patterns that is hidden.

4.3.7 Color Split Interpolation

After applying the Ramer Douglas Peuker algorithm, the number of data points are reduced which are used to draw the simplified curve. But one of the issue because of the simplification, the time information is now available at wide interval which does not produce a smooth interpolation of color. Hence to produce a smooth interpolation of color, the line segment connecting two points is checked to see the time difference between the two data points. If the time difference is significant, then the line segment is split into multiple small segment with the time divided equally between them. The split of the line segment is made instantaneously and hence there is no storage required and produces the result without additional overhead.

4.3.8 Overall Logic Implementation

The flowchart in Fig. 4.16 summarize the overall implementation explain in section previously. The original dataset is put into a serious of algorithm implementation for outlier treatment and data reduction algorithm such as Ramer Douglas Peuker algorithm. The output received after data reduction algorithm is used to represent the spatial information overlay on Leaflet map. The temporal information obtained from the result dataset is allowed to interpolate using different colors corresponding to the time. Finally the output visualization is added with interaction element that will help the user to enable easy understanding and quickly gain insights.

4.4 Custom Labels

The dataset contained information about several taxi and hence visualization of such large dataset requires a way to represent and identify every taxi. Custom labels are added to the visualization to identify the individual taxi. Along the route of each taxi, the labels are placed at certain location which helps the user to trace the route

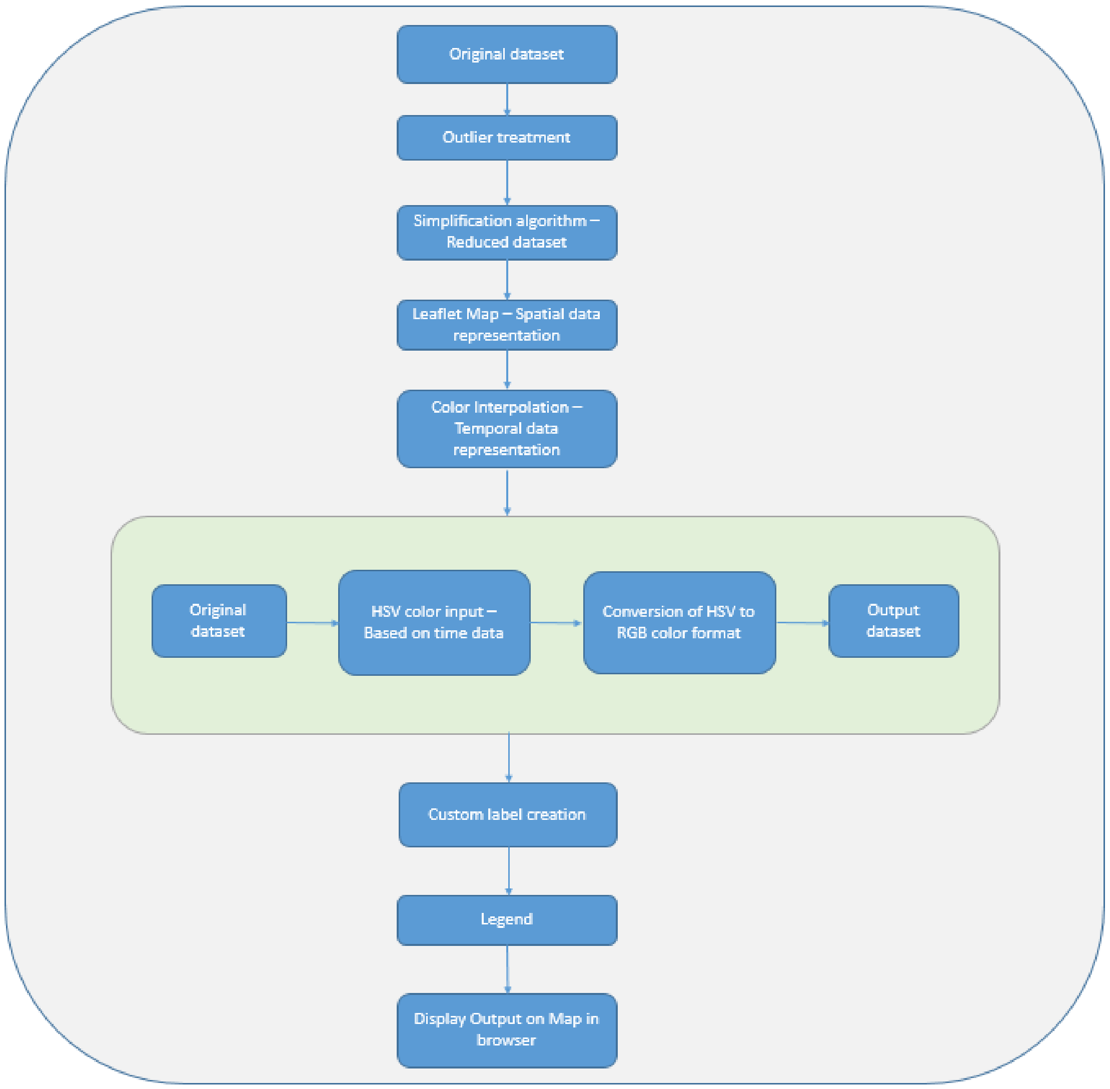

Original dataset
Outlier treatment
Simplification algorithm – Reduced dataset
Leaflet Map – Spatial data representation
Color Interpolation – Temporal data representation
Original dataset
HSV color input – Based on time data
Conversion of HSV to RGB color format
Output dataset
Custom label creation
Legend
Display Output on Map in browser

Fig. 4.16. Overall logic implementation

of individual taxi. The frequency of the placement of labels are determined based on the size of the dataset, as excess of labels results in the cluttering of the visualization.

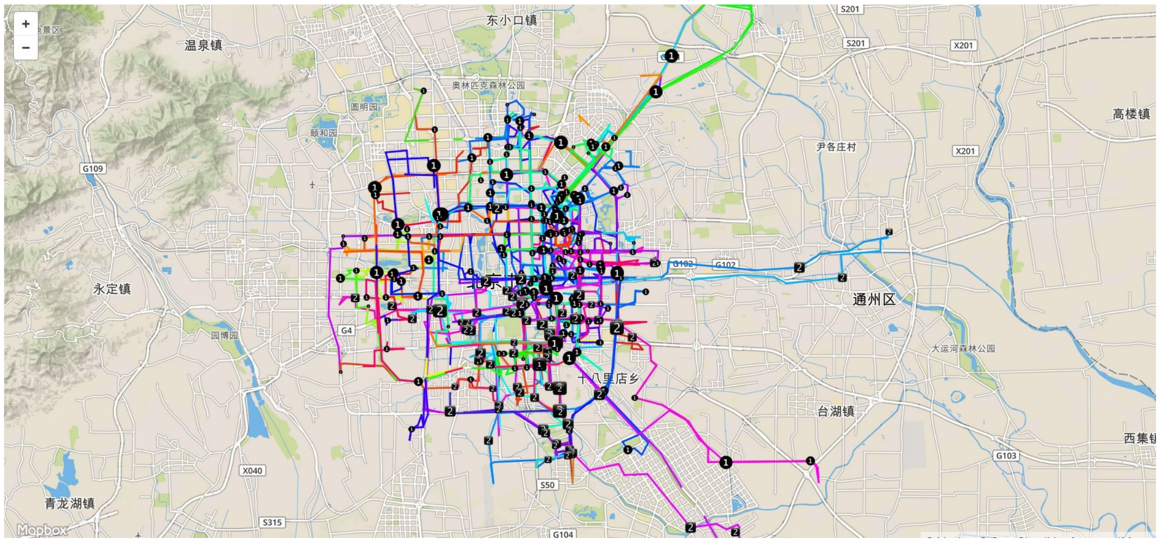


Fig. 4.17. Custom label on Leaflet

4.4.1 Size Of Labels To Represent Speed

The size of the labels depends on the speed in which the taxi is traveling at that instant of time. The greater the size of the label, the higher the speed of the taxi traveling during that instant of time and similarly the smaller the size of the label, the lower the speed of the taxi at that instant of time. Hence the variation of the size of the label helps the user to get an idea of overall speed of the taxis with respect to the location and time.

For example, one of the simple insights observed is that, taxis travel with higher speed on the highway and outer region of the city, while lower speed is observed in the busy streets of the city. It is easier to identify the streets with lower speed condition at the certain duration of the day which helps the taxi drivers to take an alternative route during that duration of the day.

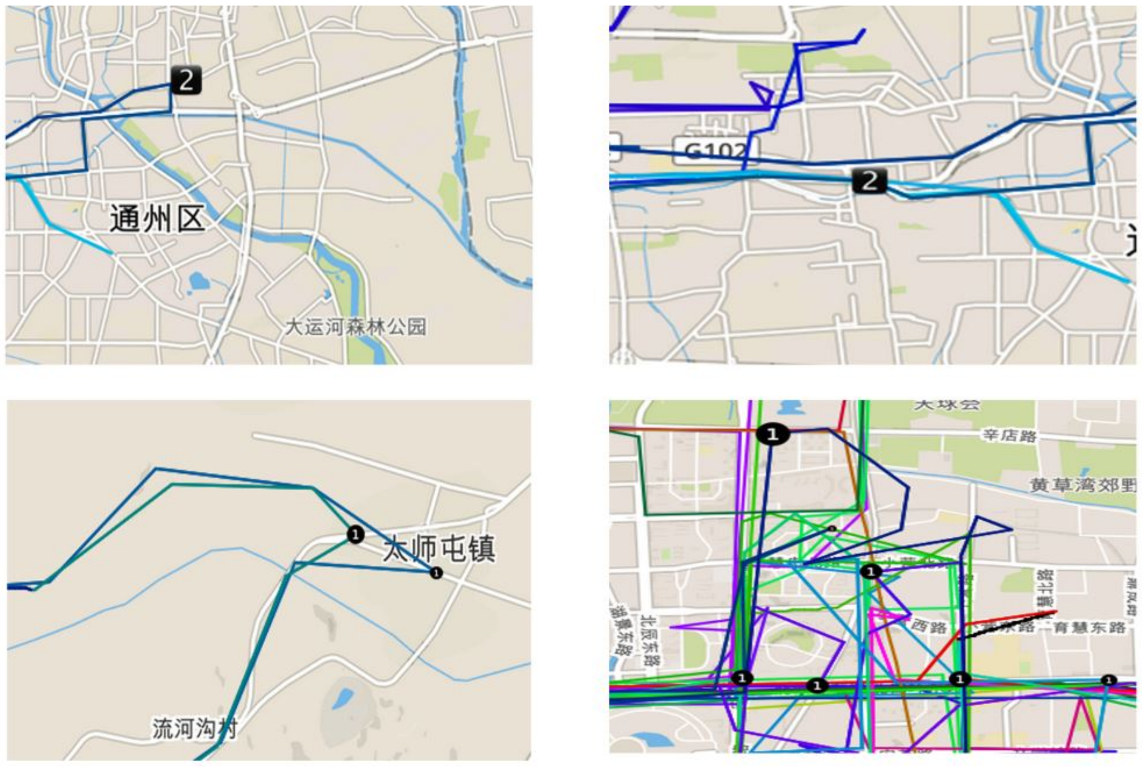
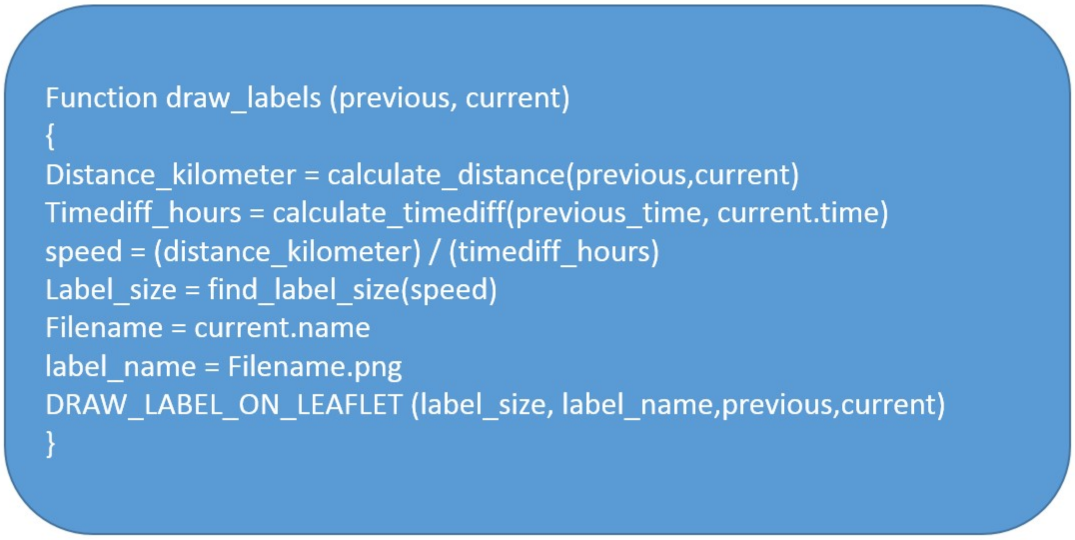


Fig. 4.18. Variation of different size of labels to represent speed

4.4.2 Addition Of Custom Labels

In the leaflet map, the labels are added on the specified point using the feature named markers as shown in Fig. 4.17. The first step is to find the size of the label for the specified point and the specified taxi. The size of the label is determined based on the speed of the taxi traveled at that instant of time as discussed previously.

The speed is calculated by evaluating the division of the distance traveled by the current point compared to the previous point in Kilometer and the time difference in hours between the current point and the previous point. The resultant speed is derived in Kilometers per hour. Fig. 4.18 depicts the variation of different size of labels to represent the speed. The bigger the size of the label, higher the speed



```
Function draw_labels (previous, current)
{
Distance_kilometer = calculate_distance(previous,current)
Timediff_hours = calculate_timediff(previous_time, current.time)
speed = (distance_kilometer) / (timediff_hours)
Label_size = find_label_size(speed)
Filename = current.name
label_name = Filename.png
DRAW_LABEL_ON_LEAFLET (label_size, label_name,previous,current)
}
```

Fig. 4.19. Pseudo code to draw label on Leaflet

traveled by the taxi. Similarly, smaller size of the label indicates slow travel by the taxi.

Once the size of the label is determined along with the file for the label, the label is created on the leaflet map by calling the appropriate function. The location for the label placement is given as the input to the marker function of leaflet. Then the marker is added as the layer to the leaflet map. The pseudo code to add the marker to the leaflet map is explained in Fig. 4.19 and Fig. 4.20.

4.4.3 User Interaction

User interaction is added to the addition of the labels to the map. As the label helps the user to trace the route of the individual taxi, it also get in the way of observing the color interpolation on the map. Hence two buttons are added to the page which enables the user to press the button to turn the label on or to turn the label off.

```
Function draw_label_on_leaflet (label_size, label_name, previous, current)
{
  var icon_anchor=icon_size/2;
  var LeafIcon = L.Icon.extend({
  options: {
              iconSize:    [icon_size, icon_size],
              iconAnchor:  [icon_anchor, icon_anchor],
              popupAnchor: [0,0]
  }
  });
var greenIcon = new LeafIcon({iconUrl: file_name});
var markers_temp=new L.marker([first.lat,first.lon], {icon: greenIcon}).bindPopup("Speed
: "+first.time+ " , "+first.date+" , "+ timediff);//.addTo(map);
map.addLayer(markers_temp);
markers.push(markers_temp);
}
```

Fig. 4.20. Pseudo code to draw label on Leaflet

4.5 Color Legend

Different colors are used in the visualization which represents the temporal dimension on the map. The user need to understand the corresponding time for the different colors to get a better perspective about the data. The user need to understand the date and time instantaneously when he/she looks into the legend. Hence in our work, we created a color legend to represent all the colors for the time interval used in the visualization.

4.5.1 Implementation Of Color Legend

Color legend is implemented using Three.js, a API in JavaScript used to create the 3D computer graphics. The first step is to set up the basic display with three.js for which we require a scene, camera and a renderer. The scene is the basic element to display anything on the screen. Any object needs to be displayed on the screen needs to be added to the scene. The camera is set to the position desired and rendered using the WebGL-renderer. Div element is created dynamically during the creation of this geometry and appended as the child to the body of the HTML. A group is a new 3D object created to add the 3D plane geometry to the group. Group is added to the scene.

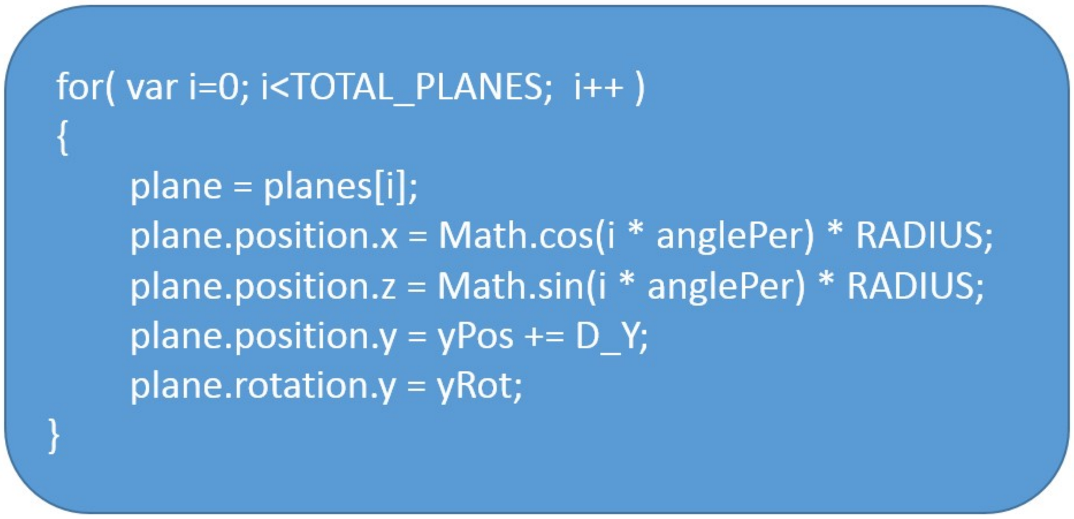
Planes are the 3D geometry created using the basic mesh material with the specific color. Number of planes created will depend on the number of rotation required for the color legend to represent. The total number of planes are estimated dynamically based on the number of time periods in the input dataset to represent the color. To get the smooth interpolation between the different color represented in the legend, the planes per rotation metric is utilized. Ideally, in our case, we use plane per rotation as 250. Hence the total number of planes will be calculated using the formula given below.

Another important parameter to draw the color legend spiral is the radius. The radius will be very essential in such as way to allow the user to visualize the smooth interpolation between the different colors in the legend. In this figure, we used 1000 as the input for the radius to the spiral. Once we know the total number of planes, in the loop, we create each planes by giving the color it needs to be displayed. The value of the hue, saturation and value is calculated and converted to the RGB color using the TO_RGB function. The color obtained as the result is given as the input metric for each planed created. Each planed is then added to the group.

The planes created from the previous step needs to be positioned based on the structure of the spiral. The x, y and z coordinates needs to be estimated for every plane to arrange them in the shape of the plane.

4.5.2 Addition Of Label To Color Legend

Color legend contains number of rotation based on the time period in the input dataset. It is dynamically calculated based on the time period which can be the



```
for( var i=0; i<TOTAL_PLANES;  i++ )
{
      plane = planes[i];
      plane.position.x = Math.cos(i * anglePer) * RADIUS;
      plane.position.z = Math.sin(i * anglePer) * RADIUS;
      plane.position.y = yPos += D_Y;
      plane.rotation.y = yRot;
}
```

Fig. 4.21. Addition of legend - Program in JavaScript

days, hours, week, month etc. The user would want to identify every rotation corresponding to the time period it represented. Hence a label is added to the side of each spiral rotation to help the user identify the corresponding time period. The text is added as 2D text document using the dynamic creation of division element (document.createElement(div)). The styles are added to the text by giving the color and the size of the text. Finally, the position of the label need to be set for the text to be placed near the start of every spiral rotation.

4.5.3 Identify Time In The Color Legend

The color legend helps the user to identify the different colors in the different time period. But it will help them more, if they were able to identify the particular color in the map and quickly able to retrieve the corresponding color in the legend along with its time. Hence when the user clicks on any of the label in the map, a small circle geometry rotated along the spiral to its position and displays the time.

4.6 User Interaction

User interactions are added to the visualization to increase the interaction among the users with the data that helps them to understand the data. It also allows the users to identify the patterns and insights from the visualization. Additionally, the user interface assist the user to narrow down to the subset of the data that they are interested instead of analyzing the entire data. It helps the user to get a detailed look on the data they wanted to perceive. Below are some of the user interaction added to the visualization.

4.6.1 Zoom In And Zoom Out

Zoom in and zoom out is one of the very important interaction available for visualization. The spatial data is represented using map which comprises of a large area of land. We need to locate to a specific region to understand the data underlying to it. It helps to resolve the overall complication of the data by narrowing it down to a specific subset of data the user is looking for.

Zoom in to a smaller geographic area in the leaflet map helps the user to look through the individual route with the corresponding color. Every individual route would correspond to a different taxi and hence help the user to trace the route of a single taxi too. Additionally the user would get an idea of the data split by different streets. The speed traveled by taxi in different street are easily monitored by zooming to the specific location.

In the implementation of the leaflet map, it is essential to set the zoom level for the output display of the map. In this experiment, the zoom level in the output is initialized to level 10. Furthermore, the user can play with the positive (+) sign and negative (-) available in the top left corner of the map as shown in Fig. 4.22.

The positive (+) sign allows the map to be zoomed in, which internally the level of zoom is increased greater than 10. The maximum allowed zoom level is set to 54 which implies that user cannot zoom greater than level 54. In the similar fashion, the

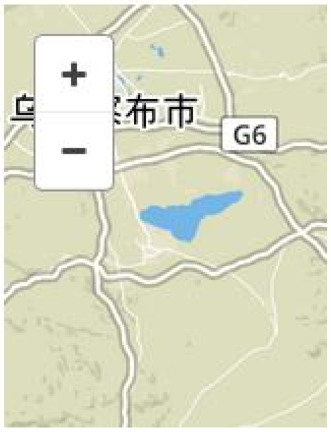


Fig. 4.22. Different zoom level drawn on Leaflet map

negative (-) allows the map to be zoomed out from the current zoom position. It is evident that internally the level of zoom is decreased lesser than current zoom level. The minimum zoom level is set to 5 which implies that user cannot zoom out lesser than zoom level 5. The various zoom level in the output visualization is shown in Fig. 4.23.

4.6.2 Label Click

There are different colors being represented in this visualization as colors represent the temporal data. When the output visualization represents multiple color interpolation for the changing time data, user need to understand the relationship between the color and the corresponding time. Color legend helps the user for such purpose. Additionally, if the user locates any specific color on the map and wanted to know the corresponding time, then the labels are used to serve the purpose. In the event of user clicking the label for the specific route, then the sphere in the color legend will locate the same color and display the time in the text box below the color legend.

In the Fig. 4.24 we can observe the different labels represented on the leaflet map. If the user wants to find the data and time respective to a single color, then the label is clicked to get the temporal information easily. When the label is clicked, the

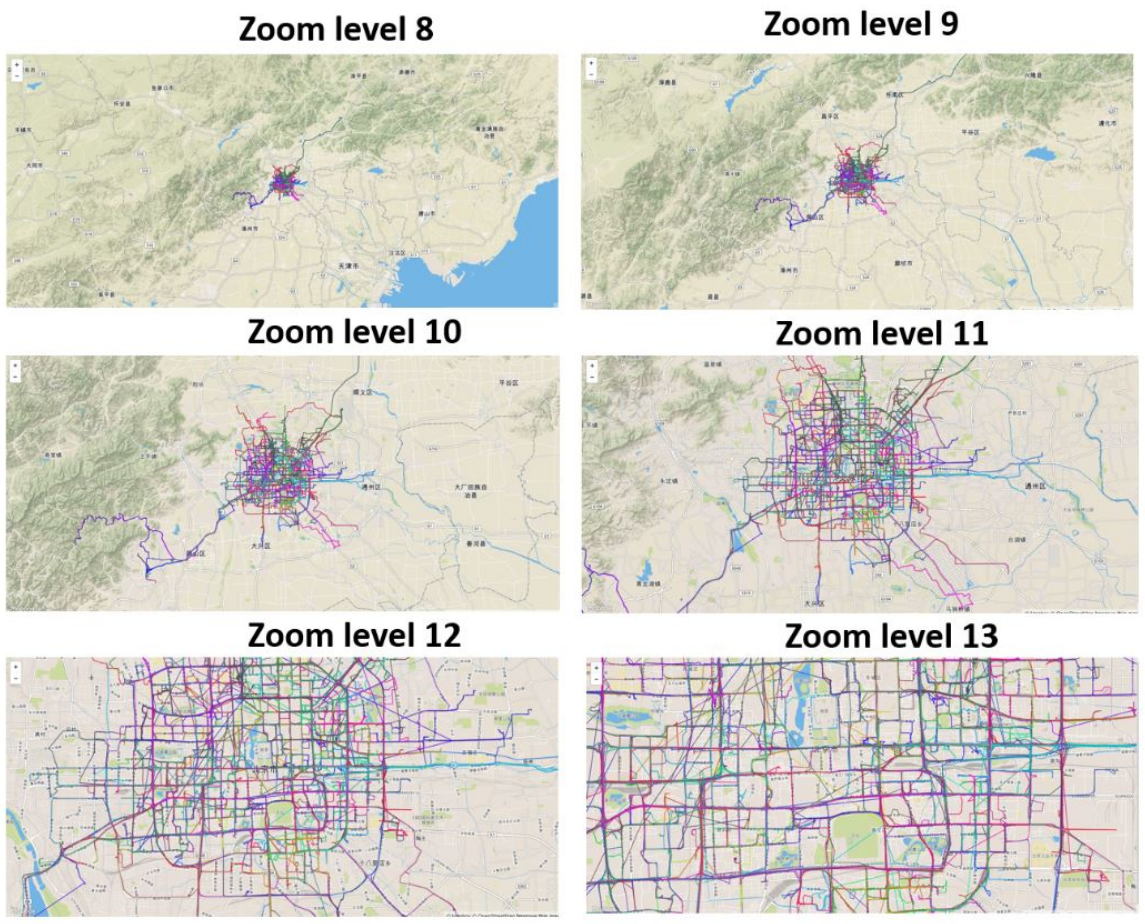


Fig. 4.23. Different zoom level drawn on Leaflet map

time corresponding to click position is obtained by the circle pointed out in the color legend. Additionally, The text box bottom to the color legend displays the time and data specific to the label click position.

4.6.3 Label Click Information

The labels are overlayed on the visualization to provide additional information for the users to help understand the data more easily. The labels helps the user to identify the taxi information. Additionally, the size of the label also helps the user

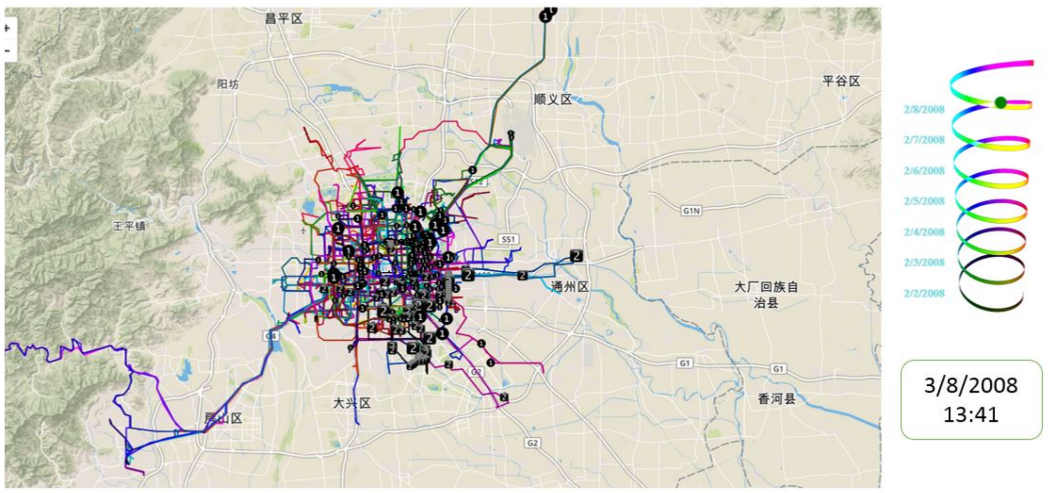


Fig. 4.24. Label Click

to relate the actual speed limit that the taxi is traveling. Furthermore, the label also contribute more help by allowing the user to click on the label to get further details. When the user click on the label, the date and time corresponding to that data point is revealed. Moreover, the speed of that data point is also displayed along with the temporal information.

In the Fig. 4.25 when the label corresponding to the taxi-1 is clicked, the information about that data point is displayed. The data and time that relates the color interpolation is revealed along with the speed the taxi has traveled. The same information can be obtained for any of the label that relates to any taxi.

4.6.4 Turning Labels On and Off

It is mentioned in the previous section that labels are displayed on the map to provide additional information regarding the temporal data and speed. But if the goal of any user to identify the trends from the color interpolation along with spatial

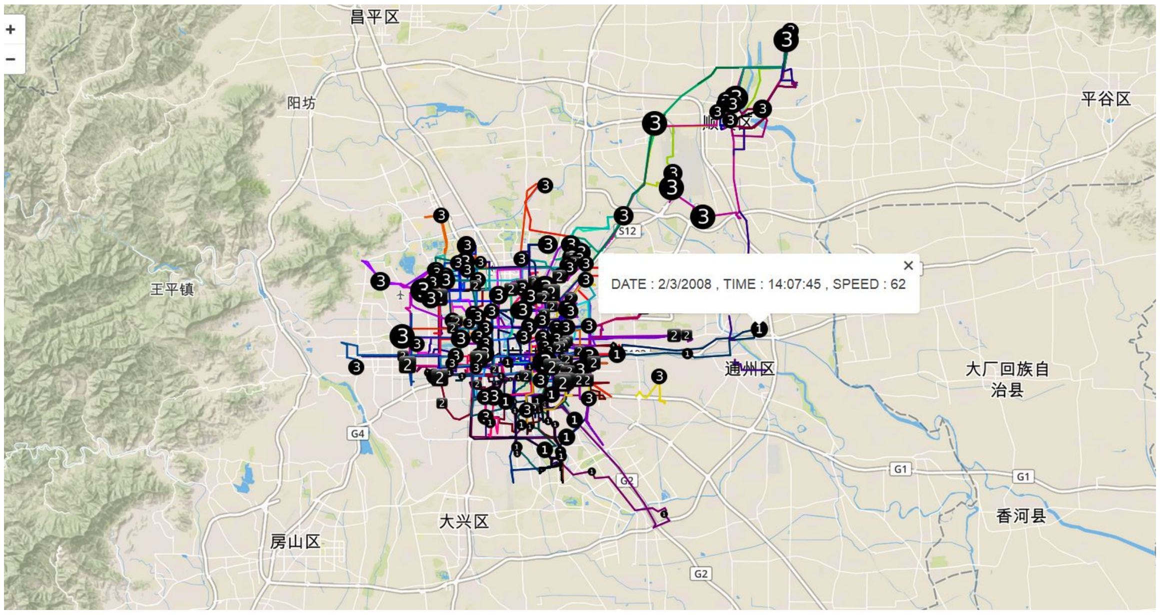


Fig. 4.25. Label click information display

information, then the labels will obstruct the path. Hence an option is added to customize the existence of the labels on the map. The two buttons namely "Icon ON" and "Icon OFF" are added to the overall screen. The default view is that labels are not added to the map. When the user press the "Icon ON" option, the labels are displayed. When the user again presses the "Icon OFF" option, then the labels will disappear from the screen. Fig. 4.26 displays the buttons added to the visualization.

4.7 Results And Observation

Various experiments are conducted on the visualization using multiple dataset in this section. It explains the different insights obtained and the knowledge gained by the user through the process.

Fig. 4.26. Icon ON and Icon OFF buttons

4.7.1 Experiment And Insights

In the first experiment, three different taxi information is considered with taxi id as 366, 534 and 650. The total number of rows were 38,527 and the total size of the dataset is reduced by applying the outlier treatment and the simplification algorithm. The resultant dataset is used to display the output visualization technique with the spatial data represented in the leaflet map and the temporal data being represented as the color interpolation among the different data points as shown Fig. 4.27 and Fig. 4.28 with different zoom level.

In the Figure 4.29, the flowchart is used to represent the step by step reduction in the overall size of the dataset. The initial size of the raw dataset for the three taxi routes contained 38,527 rows. The outlier treatment reduced the total size of the data to 27,024 which is equivalent to 70% of the total 100% raw dataset. The dataset obtained after the treatment of outliers is sent as the input to the simplification algorithm which results in 1935 rows, that is equivalent to 5% of the total 100% raw dataset.

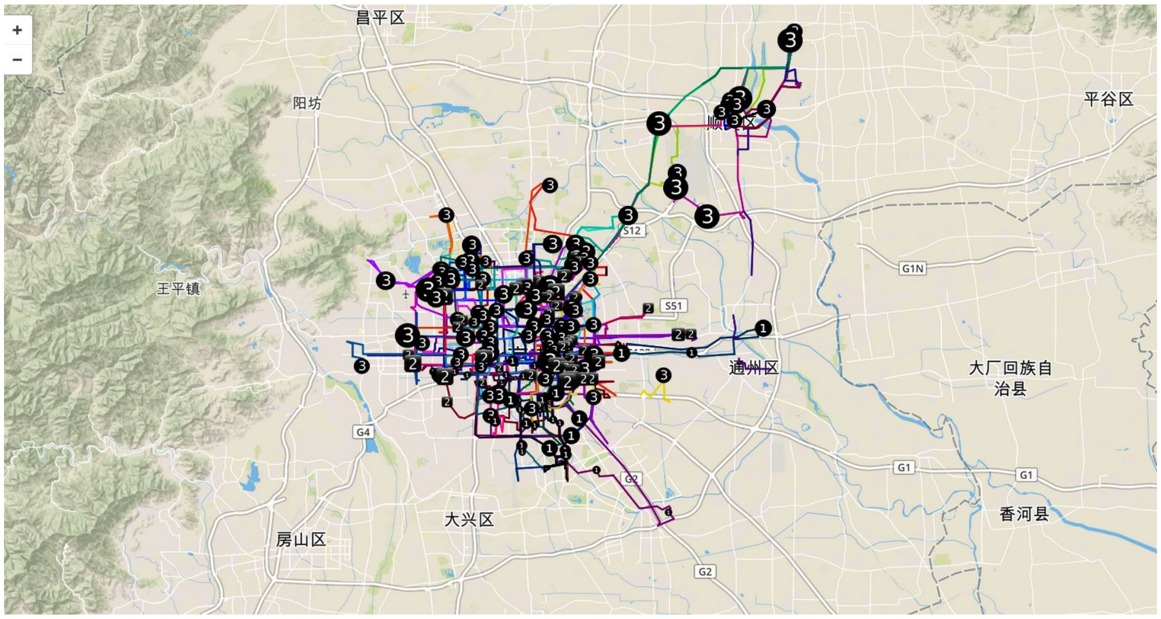


Fig. 4.27. Experiment - 1 at zoom level 10

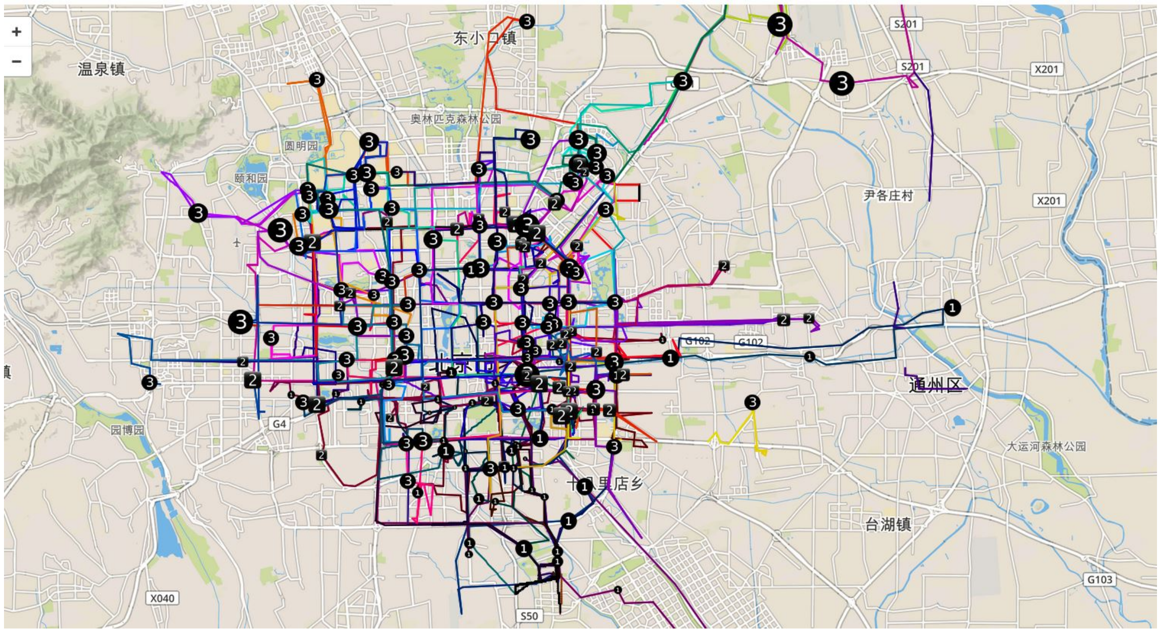


Fig. 4.28. Experiment - 1 at zoom level 11

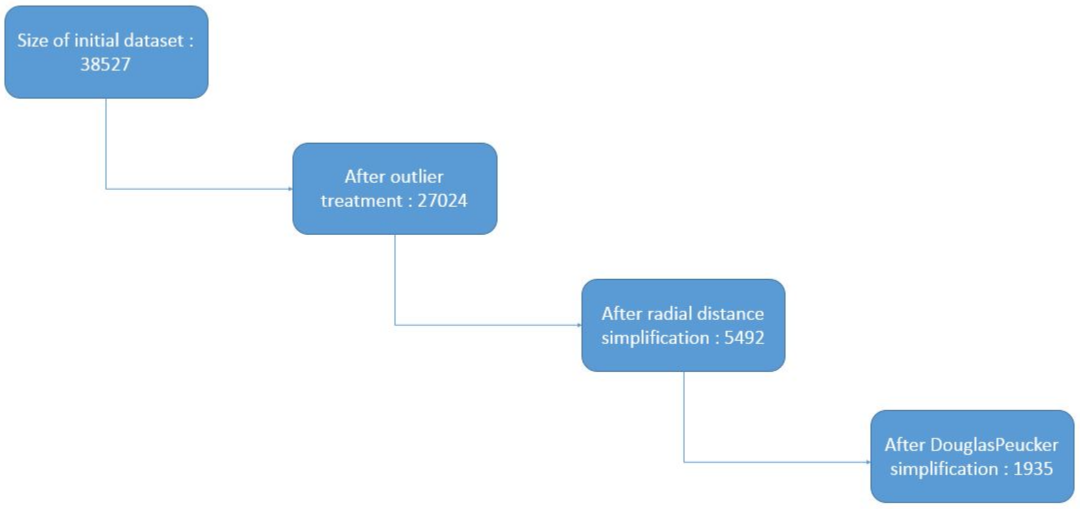

Size of initial dataset : 38527
After outlier treatment : 27024
After radial distance simplification : 5492
After DouglasPeucker simplification : 1935

Fig. 4.29. Step by step process of data reduction

The inputs to the data reduction algorithm is listed below.

- Epsilon Value : 0.002 [Used for the simplification algorithm]
- Threshold value :30 [Maximum meters by the second used for outlier treatment]
- Current Zoom level : 10 [Minimum is 5 and Maximum is 50]

Insight - 1

From the Fig. 4.30 it is clearly evident that the taxi labeled as 3 that represents the taxi named as 650. The taxi three (3) have traveled outside the city area away from the most population. This gives the user an idea of how a certain taxi have been traveling in certain area during a specific time. This taxi has traveled to the destination and back to the city area. In addition to that, the taxi has traveled to the destination in green color which represents the 10 AM in the morning and returned in the purple color which represents night time of 10 PM. Hence the user can easily identify which taxi have traveled to the far destination and what time of the day the travel has been made.

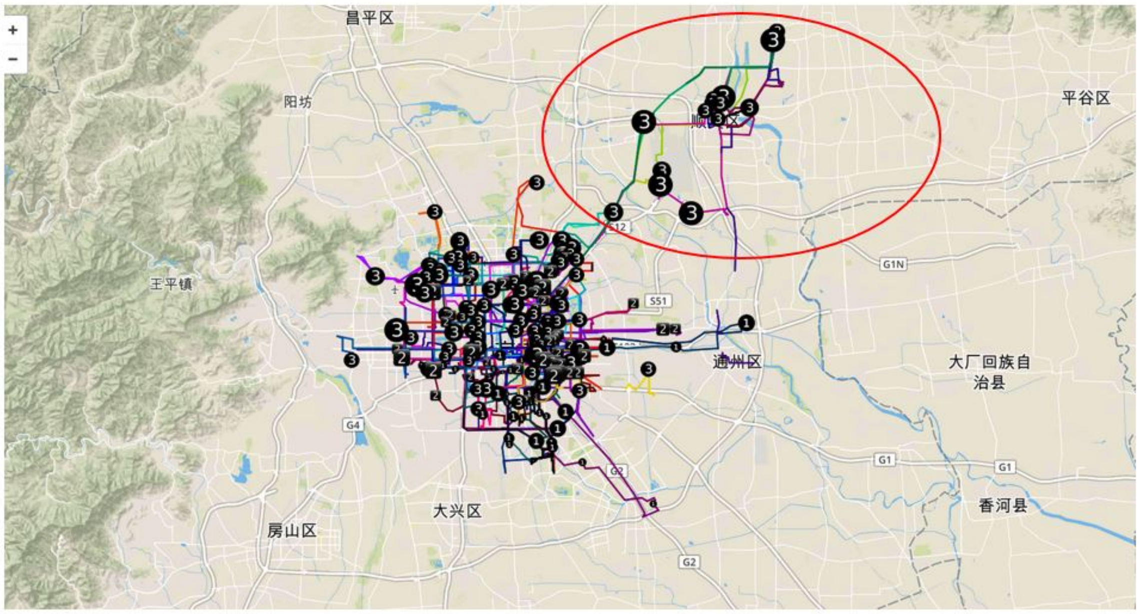


Fig. 4.30. Insight - 1 Taxi 3 travel route

Insight - 2

From the Fig. 4.31 it is clearly evident that the taxi labeled as 1 that represents the taxi named as 366. The taxi one (1) have traveled south of the city region and it is the only taxi that has gone south among the two other taxi data. From this Fig 4.31., the user can understand that three taxi have their own route and region covered during the week. Among them, taxi one have taken the south route and traveled in lower speed compared to the other regions. When the user clicks on any of the icon, they can get additional information such as the speed they are traveling and the date and time details particular to that data point. When the user clicks one of the icon in the highlighted region, it is confirmed that the speed the taxi one have traveled is 20 kilometer per hour. This proves that user can easily use the visualization to understand the route taken by any taxi and the speed information.

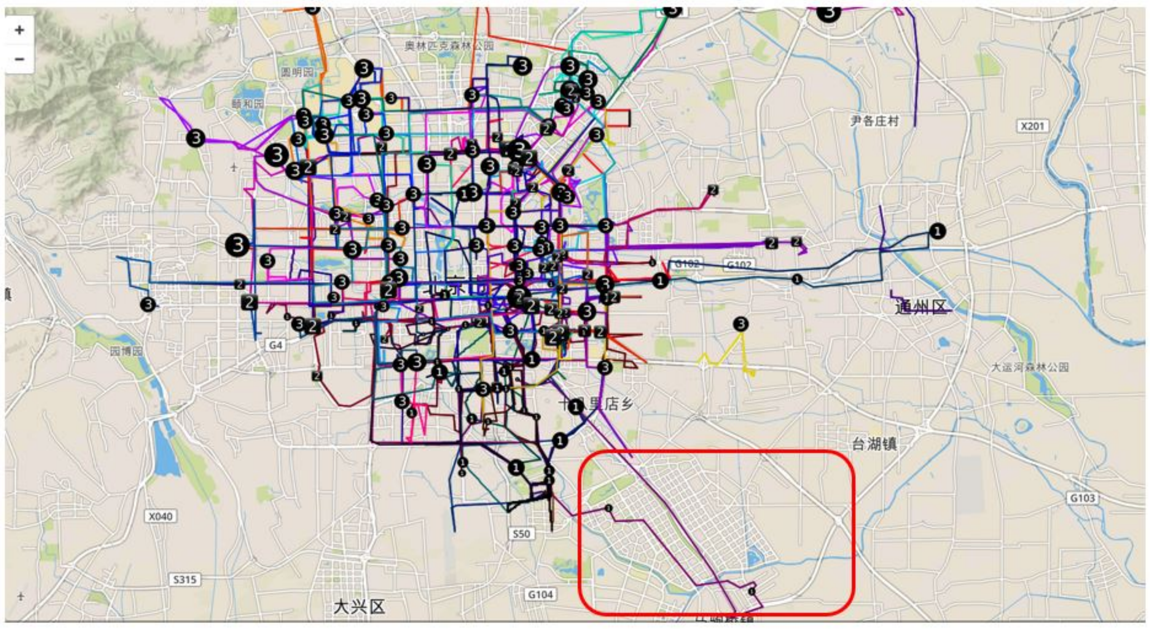


Fig. 4.31. Insight 2

Insight - 3

From the Fig. 4.32 it is clearly evident that the taxi labeled as 1 that represents the taxi named as 366. The taxi two (2) have traveled east of the city region. From the highlight region in the Fig. 4.32 the user can understand that the taxi have traveled in the purple color which corresponds to the night time. When the user actually click on any of the icon in the highlighted region, they can identify that the taxi has made the travel to and from the destination within the short time frame at night time 10.30.

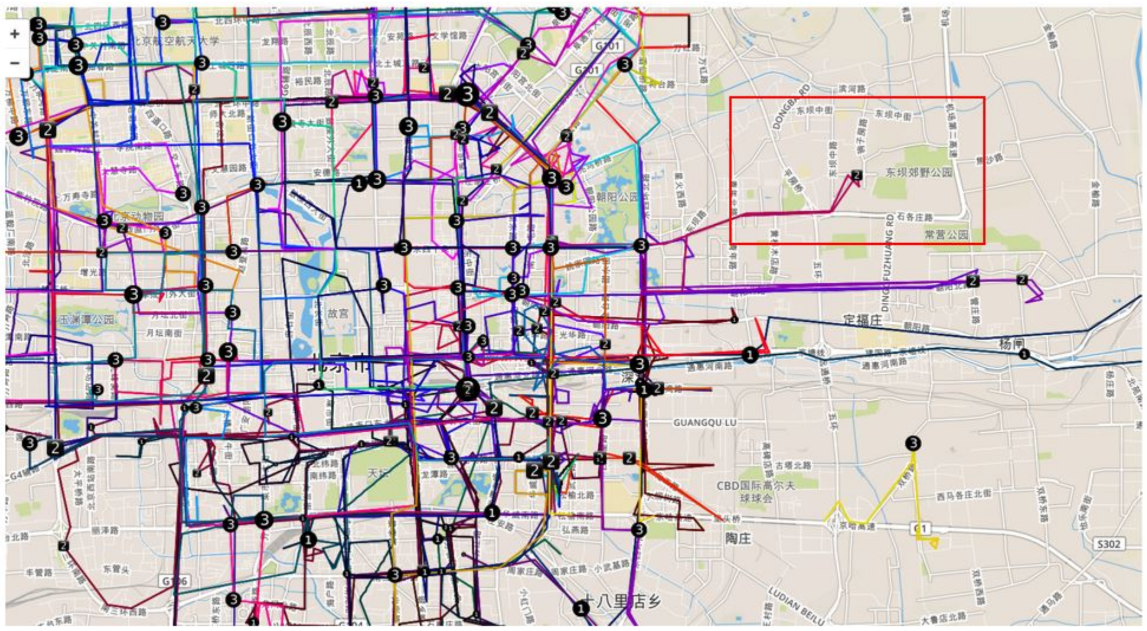


Fig. 4.32. Insight 3

Insight - 4

From the figure Fig. 4.33 the user can identify that there are lot of green color in the highlighted area of the geographic map. The time corresponding to that highlighted area can be obtained by clicking on the labels. The time falls between 6.30 A.M and 8 A.M. Evidently, the taxi traveled in the early morning time period in the highlighted region.

Insight - 5

From the figure Fig. 4.35 patterns are observed in the highlighted region. Multiple taxi have traveled in the same route many times as repetition of blue routes are observed in that region. The blue color indicates the evening time period between 5 P.M and 7 P.M. Hence the user can conclude that lot of evening traffic have been

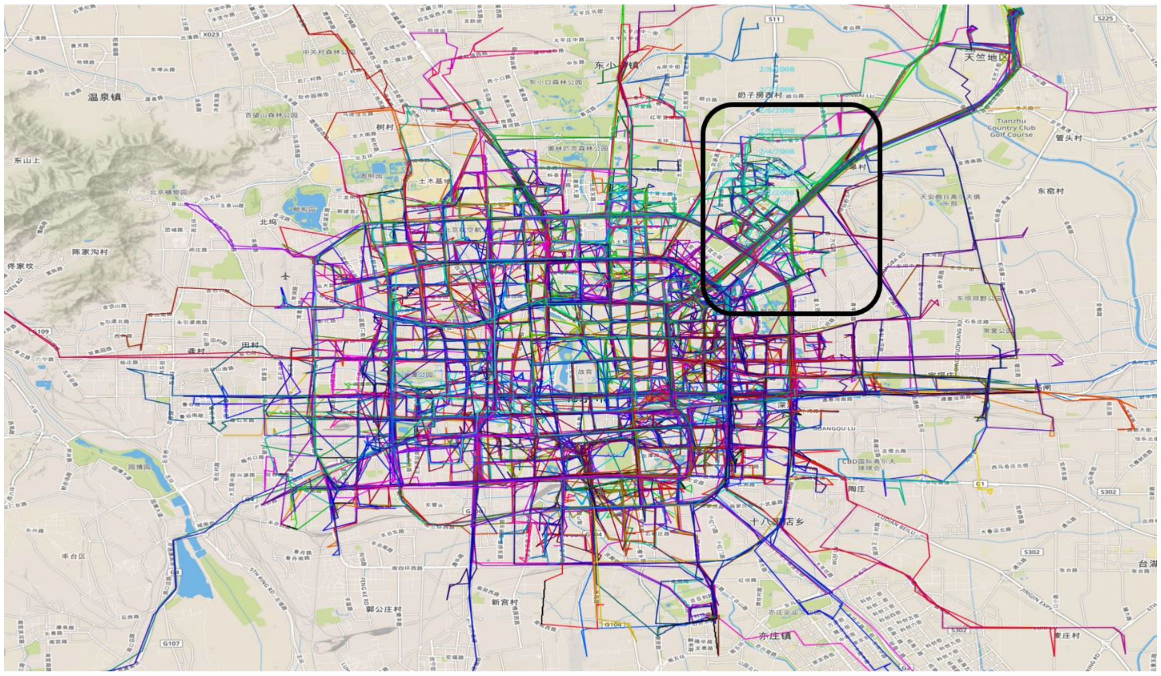


Fig. 4.33. Insight 4

focused on the highlighted region. These insights and trends helps the users to observe the traffic in the city at various time periods.

4.7.2 Performance

The performance of the visualization to display is discussed in this section. The performance is measured in the computer of configuration [configuration].

Experiment - 1

The first experiment is conducted to measure the time taken to load the visualization in the computer with the configuration mentioned in the previous section. The significance of the proposed technique is that it can load the output display very quickly in an efficient manner. To prove that, we conducted this experiment by tak-

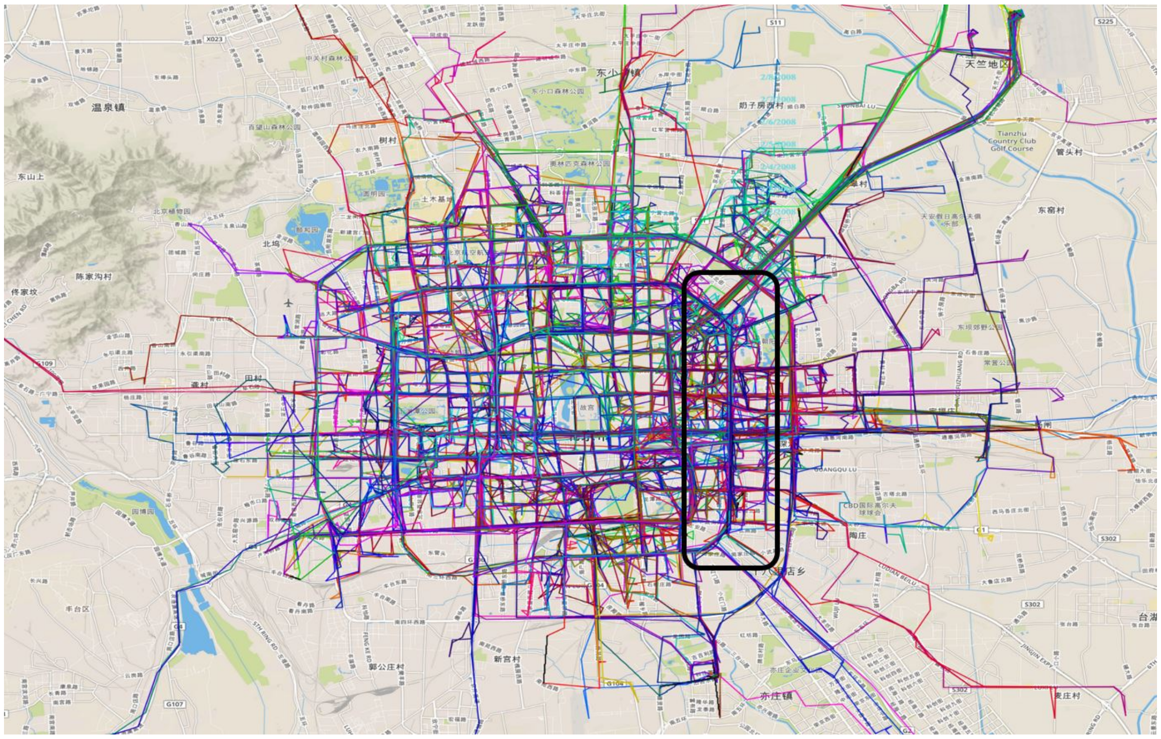


Fig. 4.34. Insight 5

ing multiple datasets of varying size. The table 4.1 shows the time taken to load the output visualization for each input dataset size.

It is evident that, as the size of the dataset is small, time taken to display the output also small. In this experiment, there are seven dataset considered with number of rows ranging from 45,714 to the maximum of 370,000. We also recorded total number of rows retained after applying the outlier removal algorithm. This output dataset is provided as the input to the input to the simplification algorithm. The output from this process results in the reduced dataset size which is used to display the output visualization.

The reduction % column tell us the percentage of reduction in the final dataset against the original dataset. The last column is the time taken to load the visualization by taking the input of the dataset obtained after applying the simplification

Table 4.2
Performance of the visualization when epsilon=0.002

Original dataset size	After_outlier treatment	After simplification	Reduction	Time taken after simplification
45714	33806	3607	92.1%	7.04
129720	93324	11219	91.4%	12.04
161868	119070	14415	91.1%	14.13
208429	162173	18710	91.0%	15.72
241091	190521	19745	91.8%	18.27
307606	244342	22645	92.6%	20.35
376000	289609	25045	93.3%	22.43
500000	390609	36119	92.8%	38.03
1000000	760609	53336	94.7%	50.43

algorithm. Fig. 4.35 visually represent the time taken in seconds for the different dataset size. The time taken steadily increases with the increase in the number of rows in the dataset is noticeable from the Fig. 4.36.

The output of the visualization has produced noticeable difference with the increase in the number of rows. The Fig. 4.31 shows the output visualization for the dataset size of 45,714 and 376,000. The dataset containing 45,714 rows take only 7 seconds to load the visualization. It contained 5 taxi routes displayed on the map and hence there is no much of visual cluttering produced. However, the dataset containing 376,000 rows take 22 seconds to load the visualization. It contained 30 taxi routes displayed on the map producing visual cluttering. Although visual cluttering is produced for the current zoom level, the user can increase in the zoom level for more accurate view of individual taxi. The icon display can be turned on to identify every route of individual taxi.

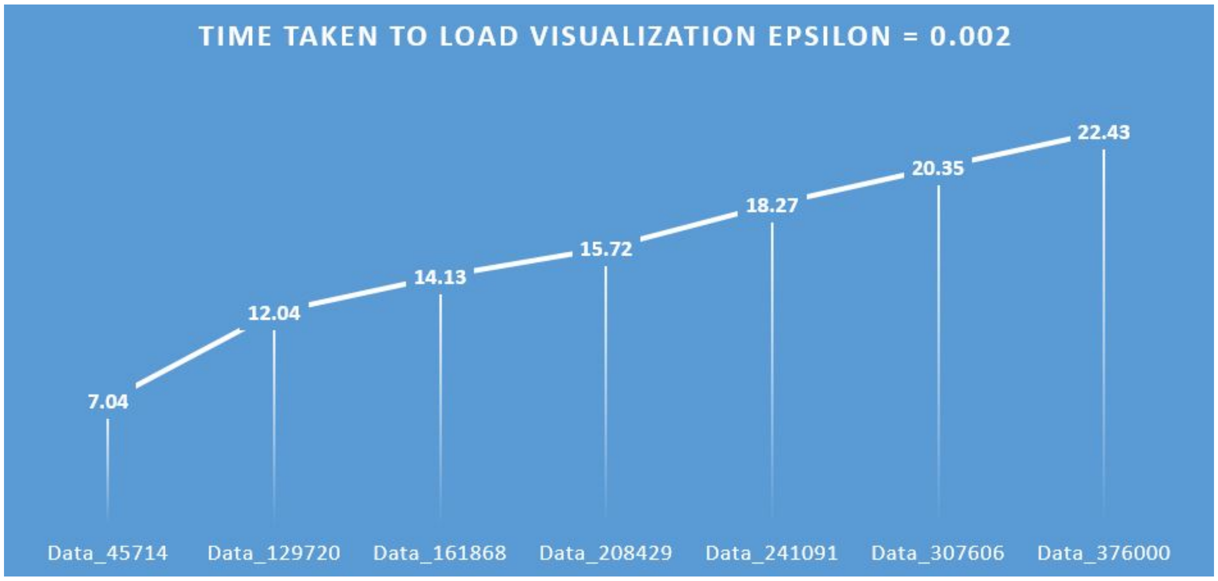


Fig. 4.35. time taken to load visualization with epsilon= 002

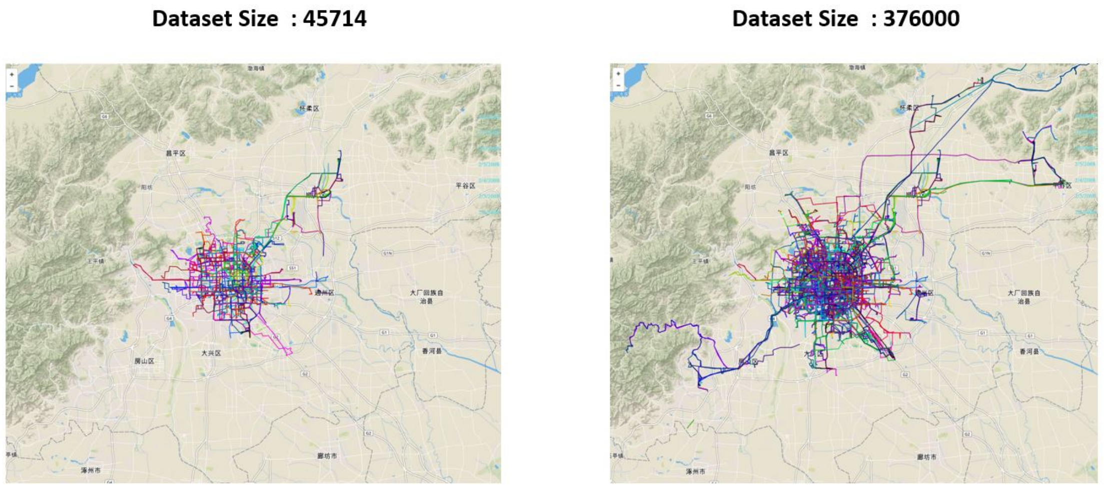


Fig. 4.36. time taken to load visualization with epsilon= 002

Experiment - 2

The second experiment is conducted to measure the time taken to load the visualization for different epsilon value. To prove that different value given for the epsilon parameter, different results will be obtained. This experiment is conducted in the same computer configuration as the previous experiment. The dataset used to handle for this experiment is also same as the previous experiment. The only change made specific for this experiment is the change in the epsilon value. Higher epsilon results in better performance but not a better output visualization. The table 4.3 shows the time taken to load the output visualization for each input dataset size.

Table 4.3
Performance of the visualization when epsilon=0.008

Original dataset size	After_outlier treatment	After simplification	Reduction	Time taken after simplification
45714	33806	1201	97.4%	3.9
129720	93324	4046	96.9%	9.48
161868	119070	5138	96.8%	11.7
208429	162173	6727	96.8%	13.89
241091	190521	7070	97.1%	17.06
307606	244342	8071	97.4%	18.95
376000	289609	8905	97.6%	19.03

It is evident that, as the size of the dataset is small, time taken to display the output also small. As discussed in the previous experiment, there are seven dataset considered with number of rows ranging from 45,714 to the maximum of 370,000 in this experiment. The number of rows that results from the outlier treatment remains the same as the result of the previous experiment. It does not vary with the change in the epsilon value as the simplification algorithm takes the input from the outlier experiment. However, the variation in the result begins with the number of rows obtained after applying the simplification algorithm. The number of rows after simplification is less when compared to the previous experiment because the epsilon value is greater than the epsilon value used in the previous experiment. In this experiment, the epsilon value is 0.008 which is greater than 0.002 used in the previous experiment.

The Reduction percentage values are higher in this experiment ranging between 96% and 98%. The percentage values are higher because in the process of simplification algorithm, only fewer data points are retained for the final visualization. For e.g. if there are 45,714 rows of data, only 1201 rows have been retained after the simplification algorithm. The remaining data has been eliminated because the epsilon value is higher and did not meet the required threshold condition.

Due to the smaller size of the output from the simplification algorithm, the visualization is loaded very quickly in the display screen. The time taken to load 45,714 of input data size is only 3.9 seconds as compared to the 7 seconds in the previous experiment. Similarly then time taken to load 376,000 of input data size is only 19 seconds as compared to the 23 seconds in the previous experiment.

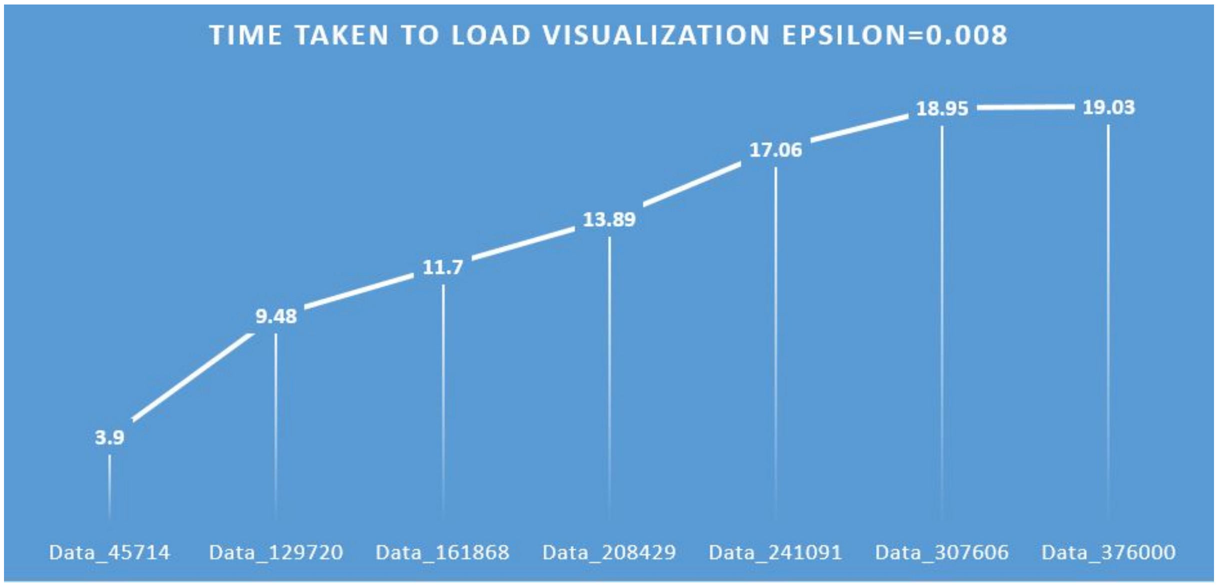


Fig. 4.37. time taken to load visualization with epsilon= 008

The output of the visualization from experiment - 2 has produced noticeable difference with the increase in the number of rows. The Fig. 4.38 shows the output visualization for epsilon = 0.008 for the dataset length of 45,714 and 376,000. The dataset containing 45,714 rows take only 3 seconds to load the visualization as it contained 5 taxi routes displayed on the map. The output visualization seems to have lost the original shape as more points had been removed from the simplification algorithm. The result hence produced is not the actual route taken by the taxi, instead the paths are interpolated to connect the points. During interpolation, the two points are connected crossing the actual roads and path in the real topology. However, the dataset containing 376,000 rows take 19 seconds to load the visualization. The lesser time compared to the experiment - 1 is attributed to the less data to visualize in the final display.

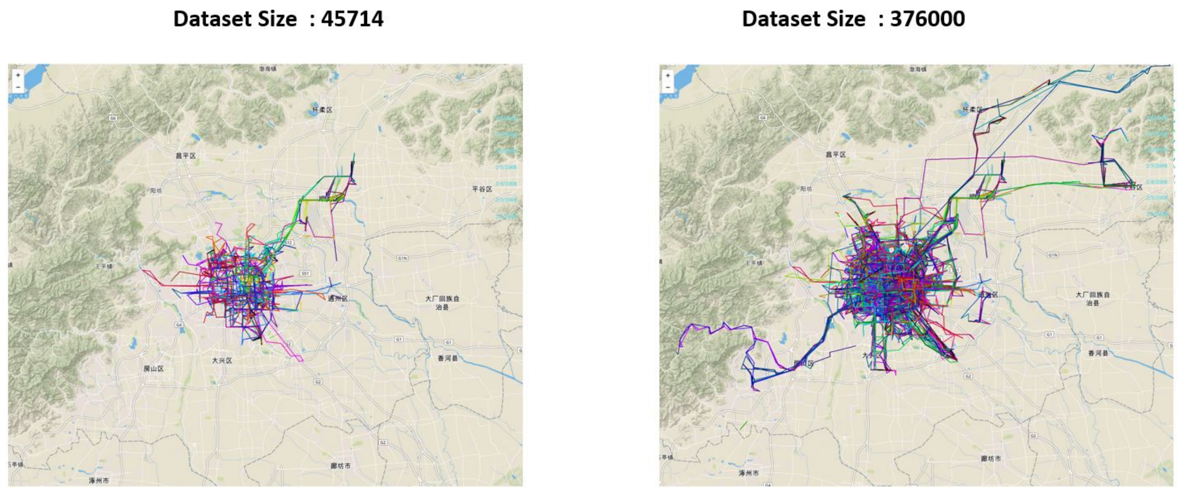


Fig. 4.38. time taken to load visualization with epsilon= 002

The Fig. 4.39 shows the graph that compares the total dataset size after applying the simplification algorithm. It shows that the first experiment with epsilon = 0.002 has greater data size compared to the experiment - 2 with epsilon = 0.008.

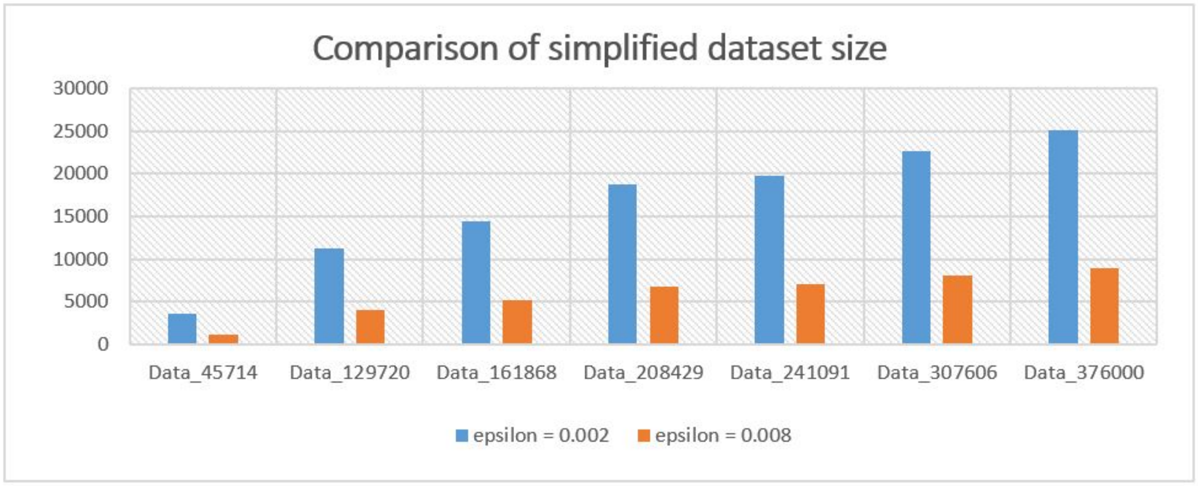


Fig. 4.39. Comparison of simplified dataset size

5. SUMMARY, CONCLUSION AND FUTURE WORK

5.1 Summary

This work is used to create a novel technique to visualize the spatio-temporal data. In this modern world, enormous amount of data are accumulated to use for later purpose like finding interesting trends and for the future improvement. Spatio-temporal data is one of the most important type of data that is used to trace the trajectory of any object or a person. The understanding of such data is very important to make judgments or awareness on the related field. Hence visualization of such large data will help to discover the data and make informed decisions. The result of this project indicate that the visualization of such large data can help the user to quickly gain insights. Also, the color interpolation will visually increase the chance of finding patterns with the data.

5.2 Conclusion

The successful completion of the implementation of such technique has been proved efficient in the various experiment conducted. From this study, the following conclusions are drawn.

1. The representation of the spatio-temporal data in two dimensional space has led to a very efficient technique and avoids visual cluttering.

2. The interaction between the spatial and temporal element has direct impact on the intensity of the insights obtained from the visualization. More accurate and information rich results are obtained from this technique.

3. We are aware that large data size will directly impact on the performance of the visualization. However, in this technique, a more efficient technique has been used to reduce the loading time and produce more interactive visualization.

4. In this technique, color is utilized to represent the temporal aspect of the data. The color interpolation provides the trajectory of the person or an object in a more intuitive method.

5. The user interaction are added to the model, which enables the user to quickly add filters and obtain the result they wanted. Zoom in will help the users to get a road wise traffic or trace the route of an individual taxi. Zoom out will help the users to get an overall trend with the routes of multiple taxi. In addition to that, the labels are added to every taxi that helps us the follow the route of a single taxi and related information.

5.3 Future Work

This visualization technique uses the HSV color model to provide the color interpolation. In the future the following can be studied.

1. In this work the color legend is implemented using three.js. The color in the visualization can be identified by clicking on the labels. It can be extended by clicking on anywhere within the visualization, to automatically get the output circle in the color legend.

2. In this visualization, the time information is uniformly distributed across the map. The speed traveled by the taxi varies between different geographic location and traffic. Hence the time distribution can be stretched to get a smooth interpolation of color based on the speed traveled.

3. The simplification algorithm have produced a simplified reduced size dataset that increases the overall efficiency of the visualization. The algorithm can

also be implemented parallelly by dividing the entire dataset into multiple taxi. This will increase the efficiency even more and decrease the loading time for visualization.

4. The visualization in this work is implemented for the stored dataset. It can be extended for the live streaming of dataset to visualize the current traffic and analyze the trends and patterns.

5. In this work the color interpolation is based on the HSV color model. In the future, the color models can be extended to the HSL or other color models to provide more depth in the knowledge of the data.

6. The path overlap is being avoided by providing the simplification algorithm. It has provided different path separately when zoomed in. The color overlapping problem could be another branch of study with the combination of colors producing a different color.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In The 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining, KDD11, New York, NY, USA, 2011. ACM.

[2] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 10, pages 99-108, New York, NY, USA,2010. ACM.

[3] Ping, Y., Xinming, T., Shengxiao, W.: Dynamic cartographic representation of Spatio-Temporal data. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, part B2, Beijing (2008)

[4] Andrienko, N., and Andrienko, G., Interactive visual tools to explore spatio-temporal variation, In M.F.Coastabile (Ed.) *Proceedings of the Working Conference on Advanced Visual Interfaces* AVI 2004, Gallipoli, Italy, May 25-28, 2004, ACM Press, 2004, pp.417-420

[5] Shrestha, A., Miller, B., Zhu, Y., Zhao, Y.: Storygraph: Extracting patterns from spatio-temporal data. In: Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, pp. 95103. ACM (2013)

[6] T. von Landesberger,, S. Bremm,, N. Andrienko,, G. Andrienko, and M. Tekusova., Visual analytics methods for categoric spatio-temporal data. In Proc. IEEE VAST, pages 183&ndash,192, 2012

[7] R. Edsall, D. Peuquet, A graphical user interface for the integration of time into GIS, Proceedings of the 1997 American Congress of Surveying and Mapping Annual Convention and Exhibition, Seattle, WA, 1997, pp. 182189.

[8] N. Andrienko, G. Andrienko, and P. Gatalsky. Visualization of spatio-temporal information in the internet. In Proceedings of 11th International Workshop on Database and Expert Systems Applications, pages 577 585, 2000.

[9] C. Tominski, P. Schulze-Wollgast, and H. Schumann. 3D information visualization for time dependent data on maps. In Proceedings. Ninth International Conference on Information Visualisation, pages 175 181, july 2005

[10] K.P. Hewagamage, M. Hirakawa and T. Ichikawa, "Interactive Visualization of Spatiotemporal Patterns Using Spirals on a Geographical Map," Proc. Symp. Visual Languages (VL ',99), 1999.

[11] Gatalsky, P., Andrienko, N., Anrienko, G.: Interactive analysis of event data using space-time cube. In: Proceedings of the Eighth International Conference on Information Visualisation (IV 2004). IEEE Computer Society (2004)

[12] M. Kraak. The space-time cube revisited from a geovisualization perspective. In Proc. 21st International Cartographic Conference, pages 19881996, 2003.

[13] Kaya, Erdem, M. Tolga Eren, Candemir Doger, and Selim Balcisoy. ”Do 3D Visualizations Fail? An Empirical Discussion on 2D and 3D Representations of the Spatio-temporal Data.” Harvard

[14] ”HSL and HSV,” Wikipedia, the free encyclopedia, accessed November 2, 2016, https://en.wikipedia.org/wiki/HSL_and_HSV

[15] ”Ramer-Douglas-Peucker algorithm,” Wikipedia, the free encyclopedia, accessed November 2, 2016, https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm