Automatic Calibration of Road Intersection Topology using Trajectories

Lisheng Zhao[1], ✉Jiali Mao[1], Min Pu[1], Guoping Liu[2], Cheqing Jin[1], Weining Qian[1],
Aoying Zhou[1], Xiang Wen[2], Runbo Hu[2], Hua Chai[2]
[1]East China Normal University [2]Didi Chuxing
[1]{51185100035, 51174500118}@stu.ecnu.edu.cn,{jlmao,cqjin,wnqian,ayzhou}@dase.ecnu.edu.cn,
[2]{liuguoping, wenxiang, hurunbo, chaihua}@didiglobal.com

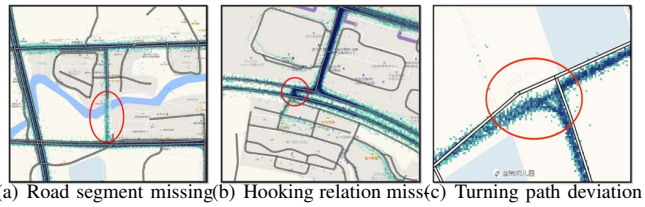
***Abstract*—The inaccuracy of road intersection in digital road map easily brings serious effects on the mobile navigation and other applications. Massive traveling trajectories of thousands of vehicles enable frequent updating of road intersection topology. In this paper, we first expand the road intersection detection issue into a topology calibration problem for *road intersection influence zone*. Distinct from the existing road intersection update methods, we not only determine the location and coverage of road intersection, but figure out incorrect or missing turning paths within whole *influence zone* based on unmatched trajectories as compared to the existing map. The important challenges of calibration issue include that trajectories are mixing with exceptional data, and road intersections are of different sizes and shapes, etc. To address above challenges, we propose a three-phase calibration framework, called *CITT*. It is composed of trajectory quality improving, core zone detection, and topology calibration within *road intersection influence zone*. From such components it can automatically obtain high quality topology of *road intersection influence zone*. Extensive experiments compared with the state-of-the-art methods using trajectory data obtained from *Didi Chuxing* and *Chicago campus shuttles* demonstrate that *CITT* method has strong stability and robustness and significantly outperforms the existing methods.**

***Index Terms*—road intersection, influence zone, core zone, quality improving, centerline fitting**

I. Introduction

As one of the world's leading transportation platform, *Didi Chuxing* (*Didi* for short) has more than 550 million registered users, and transports 10 billion passengers every year. Every day, *Didi* generates over 106 TB of vehicle trajectory data and handles more than 40 billion routing requests. In order to provide dynamic route planning for *Didi* drivers, the accuracy of road map is of vital importance. As a critical part of digital road map, road intersection is the junction of multiple interlinked roads, whose geometric and topology properties play significant role in mobile navigation and other location services. Any wrong turn at the road intersection has major influence on the drivers' safety and car-hailing platform's economic benefits. As is reported by TTG *, four British tourists were killed in a collision in Florida due of the driver making a wrong U-turn at one road intersection. For *Didi*, most of order cancellations and drivers' detour complaints were caused by incorrect or missing turning paths within road intersection area. As a result, it necessitates a stable and robust calibration mechanism to keep the road intersection topology up-to-date. Recently, road intersection detection using GPS traces has become one of the most active research in map update field, due of its high precision and low cost. Most of the existing methods [1]–[10] lay emphasis on identifying the location and coverage of road intersection, as well as the turning rule within the coverage. While they focus less on diverse spatial connectivity and complex routable topology for road intersection.

Due to constant updating of road infrastructure and dynamically changing of traffic situations, road intersection often exists incorrect geometric connectivity and imprecise allowable heading directions relative to vehicles' traveling trajectories. Fig. 1 illustrates some examples about incomplete or imprecise connection information (highlighted in red circles) at the road intersection, pale gray line represents the intersection topology recorded in the existing map, and blue point straps represent driving traces of the vehicles. As can be seen from Fig. 1 (a) and (b), a new built or reopened road segment near the cross-roads, and a hooking relation between two roads are not added into the map, which lead to the lack of allowable heading direction of road intersection. In addition, the turning centerline of the roadway connecting the junction cannot be extracted precisely, which results in the passing trajectories deviating from the existing map, as shown in Fig. 1 (c). An effective solution not only involves the identification of location and coverage, what we call the 'core zone' of road intersection, but also includes the calibration of turning path within whole road intersection influence zone (marked with green solid circle), as shown in Fig. 2.



(a) Road segment missing (b) Hooking relation missing (c) Turning path deviation

Fig. 1: Examples of incorrect connection properties of road intersection

✉ Corresponding author.

*https://www.ttgmedia.com/news/news/four-british-tourists-killed-in-florida-car-crash-13731

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00145

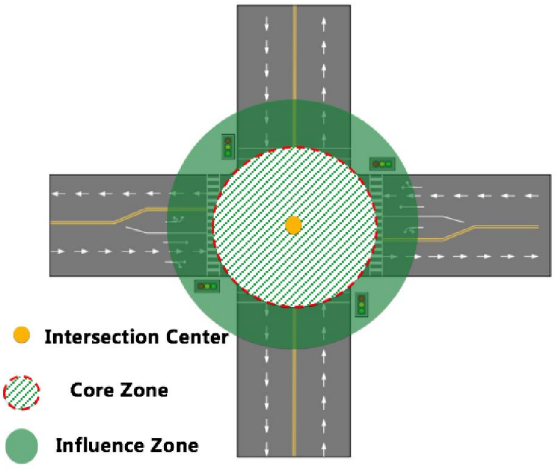


Fig. 2: Road Intersection Influence Zone

Nevertheless, topology calibration for road intersection influence zone not only faces severe challenges from uneven data distribution and low-quality data, but needs to take into account the time-varying evolving property of connectivity and similar turning behavior with nonintersection's. Specifically, trajectory distribution is highly skewed in road network. It is hard to infer the topology of region with sparse trajectories. Due of artificial shutdown or device failure of GPS sensors in vehicles, and vehicle traveling by city canyons, trajectory data is usually discontinuous and mixes with numerous location deviations. Intuitively, road intersections are of various sizes and deforms and shapes, and meanwhile the shape may change dynamically with the traffic rules or road infrastructure maintenance. What is more, non-intersections have similar turning behaviors with intersections. All of the above in turn brings the interference to our road intersection topology calibration issue.

To tackle above challenges, we propose a three-phase calibration framework for road intersection topology using trajectories, called as *CITT*. It consists of trajectory quality improving, core zone identification, and turning path extraction within influence zone. At the first phase, a quality improving strategy composing of segmentation, denoising and simplification is put forward. It can alleviate the effects of GPS signal interruption and noise data, and sharply reduce the amount of sampling points while keeping the characteristic of vehicle turning behavior in trajectories. At the second phase, based on the variation of heading direction property and speed feature of trajectories, a core zone detection mechanism consisting of location identification and coverage determination of road intersection is presented. At last but most important phase, a two-step strategy is designed to calibrate the topology for road intersection influence zone, which is comprised of turning centerline fitting and map matching. Finally, *CITT* method merges the updated topology of road intersection influence zone into the original road map. Distinct from the existing map update methods, due of the complexity of road geometry at road intersection, our topology calibration method adopts the strategy of turning centerline fitting first and detect unmatched turning path later. Note that *CITT* method is executed periodically to obtain up-to-date road intersection topology using the newest arrived trajectory data. In summary, the contributions of this paper are as follows:

- To the best of our knowledge, ours is the first work to calibrate the topology of *road intersection influence zone*. We design a calibration framework called *CITT* consisting of improving data quality, identifying *core zone* and updating road intersection topology.
- We put forward two key points to facilitate *core zone* identification. The location identification of road intersection depends on *direction-based* clustering and *speed-based* checking, then the coverage of road intersection is determined based on heading direction difference and speed variation characteristics using annulus geometry.
- We design a centerline fitting mechanism, including $Frechét-$ *based sampling*, *force-attraction adaption* and *B-Spline smoothing*, to ensure the accuracy of turning path extraction within road intersection influence zone.
- A thorough comparative experimental evaluation with the state-of-the-art methods based upon trajectory data obtained from *Didi Chuxing* and *Chicago campus shuttles* verifies the effectiveness and and efficiency of *CITT* method.

The remainder of this paper is organized as follows. In Section II, we review the latest work related to our research. In Section III, the preliminary concepts are introduced and the problem is defined formally. In Section IV, we outline and analytically study *CITT* scheme. In Section V, a series of comparative experiments are conducted on two real data sets to evaluate our proposal. Finally, in Section VI, we provide a conclusion and point out future directions.

II. Related Work

In the past decade, the issue of intersection detection has attracted wide attentions and interests of academe and industries. As vital component of road map, numerous trajectory-based map update methods ([2], [11]–[21]) employed the techniques such as clustering, trace merging and kernel density estimation, and represent the road intersection by a single vertex. However, the vertexes extracted by them usually not only contain real road intersections but include possible breakpoints on the roadways. Therefore they cannot guarantee obtaining the road intersections that are coincide well with actual road network, let alone generating the road intersections with detailed topology structures.

To proliferate the accuracy of road map, several approaches ([1], [3], [5], [6], [8], [13], [22]) focused on the vehicles' trajectories around road intersections and attempted to identify the location of intersections. These detection methods were based on the premise that majority of vehicles frequently changed driving directions at the intersections instead of on the roadways. Fathi et al. trained a classifier in the form of

shape descriptor based on sub-trajectories features to distinguish intersections from non-intersections [1]. Karagiorgou et al. presented an intersection detection method by extracting *turn samples* according to the specified speed threshold and direction threshold [13]. Wu et al. designed an intersection recognizing mechanism by *turn points* searching and *converging points* clustering [22]. Xie et al. tried to find the longest common subsequences using dynamic programming and then derived the connecting points from partitioned subsequences to identify the intersections [6]. Li et al. attended to identify the intersections through extracting dominant orientations and merging similar orientations while maintaining independent conflicting orientations [7]. In our previous work [8], a two-phase framework, called RIDF, was put forward to detect the location of road intersection. Through extracting candidate cells based on direction statistic analysis and intersection location refining using hybrid clustering strategy, our proposal could effectively detect road intersections of different sizes.

With the improvement of trajectory data quality, a few researches tried to make sense of the connectivity of road intersection and inferred coarse-grained intersection map. These works involve the location as well as boundary detection, and turning rules discovery of road intersections. Based on the assumption that curved trajectories often occur at intersections instead of on roadways, Wang et al. detected the locations and turning paths of road intersections by *turns* extraction, density grid generation and traffic rule deriving [3]. To generate detailed structural models for road intersections, Deng et al. presented a three-step approach to extract rough spatial coverage, internal turning modes and structural representations of road intersections [9].

To meet high-accuracy requirement of map navigation application, timely calibration issue of road intersection topology is still to be solved. Distinct from the aforementioned road intersection detection methods, our proposal implemented in a more delicate manner, and tried to provide detailed geometry structures for road intersection influence zone. It can be essentially regarded as a comprehensive solution for sub-road-network update problem, which is composed of data denoising, data segmentation, data simplification, location as well as boundary identification of road intersections, turning clusters differentiation, centerline fitting and map matching, etc.

III. Problem Definition

In this section, we introduce some preliminary concepts and formalize the calibration problem of road intersection topology using trajectories.

In a bid to provide accurate map service for vehicle navigation or route planning application, our aim is to detect the topology changes within the road intersection influence zone, the primary focus of which is to mark the missing turning paths or possible location deviations on the existing map.

Definition 1 (Road Map): A road map is typically represented as a graph $G_m = (V, E)$, where the set of vertexes (denoted as V) includes road intersections and other break-points on the roadways, and the set of edges (denoted as E) involves the roadways among the vertexes.

Distinct from the vertexes in traditional graph structure, any vertex V_i in G_m has its own geographic coordinate. Continuously arrived trajectory data enables the updating of the geolocations of vertexes timely.

Definition 2 (Trajectory): A trajectory Tr refers to a sequence of positional points that chronologically sampled during a time period, denoted as $Tr = \{p_1, p_2, ..., \cdots, p_n\}$, where $p_i = (lng_i, lat_i, t_i)$, lng_i as well as lat_i denote the longitude and latitude of one positional point at timestamp t_i, and $\forall i < j$, p_i arrives earlier than p_j.

Two adjacent points can be connected into a trajectory segment, thus a trajectory may also be coped with in the form of a sequence of trajectory segments. Since trajectories are collected in real-time with massive scale, we calibrate road intersection topology based on the trajectories arrived in each time window using the sliding window model. To proliferate the quality of raw trajectory data, we leverage the preprocessing steps consisting of denoising, split and simplification.

In view of the property that road intersections are of different sizes, we employ a *top-down cell-partitioning* strategy to extract different-size grid cells containing road intersections. Here the size of each cell initially is set as 200×200 and gradually split until that of cell is 25×25 by quad-tree data structure. Additionally, it is observed that the points with multiple turning directions would gather at the road intersections instead of on the roadways, and they usually need to keep lower speeds due of slowing down when cornering. Therefore, we group the trajectories within each grid cell into various classes using *direction-based* clustering method, and screen out the cells that not only have at least k ($k \geq 3$) clusters but keep the speed of not exceeding average speed of all the points, to find road intersection cells.

Definition 3 (Road Intersection Cell): Given a collection of cells $C(G)$ and a direction number threshold th_d ($th_d \geq 2$), a cell C^l ($C^l \in C(G)$) is a road intersection cell iff it contains at least th_d distinct clusters and the average speed of the points within C^l is less than that of its neighboring cells.

To determine *core zone* within each road intersection cell, we attend to identify the central location of road intersection using *Meanshift* method, and detect the coverage range of road intersection based on the variations of direction and speed features. Subsequently, we extend outward the range of *core zone* to obtain *influence zone* according to a specified radius threshold r_z. After clustering all the trajectory segments within *influence zone* into various groups in terms of heading direction property, we try to generate a smooth turning centerline for each group to obtain each turning path for supporting navigation applications.

Definition 4 (Turning Centerline): A turning centerline, denoted as Tc, is represented by a smooth contour line. It is linked by a sequence of consecutive positional points, $(p_1, p_2, \ldots, p_l)$ $(1 < l \leq n)$.

Finally, we summarize the problem as below.

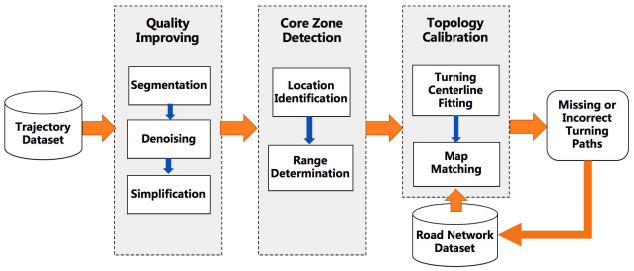


Fig. 3: Overview of Calibration Framework

Problem Definition: Given a road map G_m and a collection of continuously arrived raw trajectories, our objective is to find the missing or incorrect turning path within *road intersection influence zone* as early as possible, and based on them update the road map G_m.

IV. Overview

With the rapid development of urban infrastructure, road intersection topology always keeps physically changing due of new built (or blocked, re-opened) roadways and road intersections. In addition, traveling limits especially turning rule of road intersection within different time periods, what we called logical topology, may vary with dynamical changing of traffic situations in road networks. Continuously generated trajectories by vehicles provide us the opportunity of detecting the changes of road intersection topology. In this section, we put forward a three-phase calibration framework of road intersection topology using trajectories, called as *CITT*. As shown in Fig. 3, *CITT* is composed of trajectory quality improving, *core zone* detection and topology calibration within *influence zone*.

A. Trajectory Quality Improving

Due to signal interruptions and signal drifts caused by internal equipment issue or external human factors, the sampled positional points may be uncertain and even abnormal. Consider that low-quality trajectory may result in errors in the subsequent steps of the calibration framework, we employ the strategy consisting of segmentation, denoising and simplification to improve quality of raw trajectories, as illustrated in Algorithm 1.

Segmentation. First, there may be large sampling interval between any two consecutive positional points due of GPS signal interruption. Besides, a vehicle may enter a same road intersection more than once within a shorter time interval due of navigational error or the other reasons. The cases mentioned above make two adjacent sampling points far from each other. This increases the difficulty of road intersection identification according to the variation among the sampling points' heading directions, because the heading direction of any point is obtained by calculating the angle difference between its neighboring point and itself. To tackle this issue, we segment raw trajectory into partitions according to

Algorithm 1: Trajectories Quality Improving

Input: The raw trajectories set Tr_{raw}
Output: The trajectories set after quality improvement Tr;
1 $Tr_{segmented} \leftarrow \emptyset$;
2 **foreach** $tr \in Tr_{raw}$ **do**
3 **foreach** $p_i \in tr$ **do**
4 $timeDiff \leftarrow p_{i+1}.t_{i+1} - p_i.t_i$;
5 $distGap \leftarrow dist(p_i, p_{i+1})$;
6 // thr_{imax}: Time interval threshold
7 // thr_{dmax}: Distance gap threshold
8 **if** $timeDiff > thr_{imax}$ ***or*** $distGap > thr_{dmax}$ **then**
9 $tr_{seg} \leftarrow segmenting(tr)$;
10 $Tr_{segmented} \leftarrow Tr_{segmented} \cup tr_{seg}$;
11 **foreach** $tr_s \in Tr_{segmented}$ **do**
12 // thr_{num}: Trajectory point amount threshold
13 // thr_{len}: Trajectory length threshold
14 **if** $tr_s.amount < thr_{num}$ ***or*** $tr_s.length < thr_{len}$ **then**
15 **continue**;
16 // thr_{hmax}: Heading direction difference threshold
17 $Tr_{remove} \leftarrow removeNoisePoints(Tr_{raw}, thr_{hmax})$;
18 $Tr \leftarrow Tr \cup DouglasPeuckerReduce(tr_s)$;
19 **return** Tr;

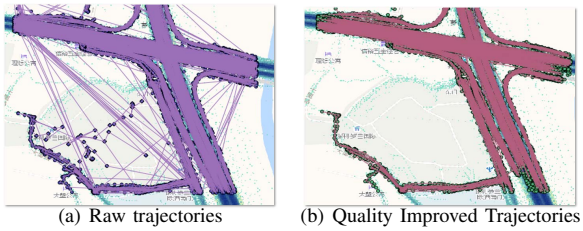
specified distance gap threshold thr_{dmax} and time interval threshold thr_{imax} (lines 1-10). Here the values of thr_{dmax} and thr_{imax} are set based on the distribution statistics of the time interval and distance gap among adjacent points. In subsequent experiments, thr_{dmax} is empirically set as 65 meters and thr_{imax} is set as 9 seconds (three times higher than the average sampling interval).

Denoising. Second, trajectory segmentation result generates a few trajectories of short length or few points, which cannot completely reflect the road geometry and adds the interference of obtaining turning paths. Therefore, such trajectories are viewed as noise data and shall be eliminated (lines 11-17). In the experiment on trajectory data set of *Shenzhen* City, we filter the trajectories if its length is shorter than thr_{len} (here thr_{len} is set as 70 meters) or the number of trajectory points is less than thr_{num} (here thr_{num} is set as 5). Additionally, GPS signal drift brings a few fluctuating trajectory points, they also shall be removed. Specifically, for any point in a trajectory, if both the direction differences between its and its pre and post adjacent points are more than 35 degrees, it would be regarded as noise point and removed from the points (line 17).

Simplification. Third, tremendous amount of sampling points are generated every hour, which leads to massive computational overheads for our calibration processing. To this end, we leverage *Douglas Peucker* based method (or *DPSIMP* for short) to simplify each trajectory into a set of key points while preserving the shape characteristics of raw trajectory (line 18). To be specific, for any trajectory tr_l, its starting point p_s and ending point p_e are connected into a line (denoted as $l(p_s, p_e)$), and the distances between each of the other points in tr_l and $l(p_s, p_e)$ are calculated in order separately. The first point p_k whose distance to the line $l(p_s, p_e)$ is greater

than the predefined distance threshold thr_{dis} (here thr_{dis} is empirically set as 5m) will be treated as one key point and retained. Then the starting point (or ending point) and the key point p_k are connected into a new line and the above distance calculation operation is repeated until the starting point (ending point) overlaps with a certain key point. Next, such key point extraction process is iteratively implemented on the rest points of tr_l. Finally, a set of key points is obtained to form a new simplified trajectory.

As exemplified in Fig. 4, the trajectories become smoother and have few noises through quality improving procedure.



(a) Raw trajectories (b) Quality Improved Trajectories

Fig. 4: An Example of Trajectory Quality Improving

B. Core Zone Identify

To detect the detailed topology within road intersection influence zone, the first step is to identify *core zone*, i.e. determine the location and coverage of road intersection using quality improved trajectories. Algorithm 2 elaborates the process of extracting the position and coverage of road intersection. Consider that road intersections are of different sizes, we first attempt to split the observing region into various sized cells by employing a top-down *quadtree*-based cell division method (lines 1-15). During the process of road intersection cell searching, we empirically set the minimum side length of *quadtree* as 25 meters, and search the road intersection cell layer by layer starting from 200 meters (i.e. the fourth layer from the bottom of *quadtree*). According to the observation that the trajectories at the intersections usually have more turning directions and lower cornering speeds than on the roadways, we implement speed analysis and direction-based clustering analysis (here *DBSCAN* [23] method is used) for all the key points within each cell to screen out the road intersection cells (Definition 3).

1) Location Identification: Road intersection is linked with multiple roadways, accordingly, the cell where road intersection exists usually contains multiple turning directions except going straight. We implement *Direction-based* clustering for all the key points within each cell and calculate the number of clusters with different turning directions.

***Direction-based* Clustering.** Due to that heading direction of each point is represented by the angle between it and true north, we cannot directly extract distinct direction cluster based on the angle difference among the points. For instance, 0° and 356° may be expressed as the same direction. To that end, we try to convert one-dimensional heading direction

Algorithm 2: Core Zone Identification

Input: The set of quality improved trajectories Tr, the cell side length l;
Output: Intersections' central location set $\mathcal{CP}$, Intersection influence zone coverage set $\mathcal{IZC}$;

1 The set of Road Intersection Cells: $insCell \leftarrow \emptyset$;
2 // Divide the entire observing region using *QuadTree* model
3 $quadTree \leftarrow QuadTree(Tr, l)$;
4 $i \leftarrow$ The current depth of the quadTree;
5 **while** $i \leq quadTree.maxdepth$ **do**
6 $\quad C^l_{temp} \leftarrow$ All cells at layer i;
7 $\quad$ **foreach** $c^l_j \in C^l_{temp}$ **do**
8 $\quad\quad c^l_j.kp \leftarrow$ key points in Cell c^l_j;
9 $\quad\quad c^l_j.dir_{num} = DirClu(c^l_j.kp)$;
10 $\quad\quad$ // thr_{dmin}: minimum number threshold of directions within the cell
11 $\quad\quad$ **if** $c^l_j.dir_{num} > thr_{dmin}$ ***and*** $c^l_j.avgSpeed < avgSpeedNeighborCell$ **then**
12 $\quad\quad\quad$ **if** $c^l_j.parent \in insCell$ **then**
13 $\quad\quad\quad\quad insCell.remove(c^l_j.parent)$;
14 $\quad\quad\quad insCell \leftarrow insCell \bigcup \{c^l_j\}$;
15 $\quad i \leftarrow i + 1$;
16 $\mathcal{P}_{cell} \leftarrow PointInCell(insCell)$;
17 $\mathcal{CP}_{temp} \leftarrow Meanshift(\mathcal{P}_{cell})$;
18 $\mathcal{CP} \leftarrow mergeRedundantPoint(\mathcal{CP}_{temp})$;
19 $\mathcal{P}_{turning} \leftarrow TuringPoint(Tr)$;
20 **foreach** $cp \in \mathcal{CP}$ **do**
21 $\quad r \leftarrow r_{min}$;
22 $\quad$ **while** $r < r_{max}$ **do**
23 $\quad\quad num_{tp} \leftarrow PointInRegion(cp, r)$;
24 $\quad\quad AVG_{cov} \leftarrow AvgSpeed(\mathcal{P}_{current})$;
25 $\quad\quad Di_{tp} \leftarrow densityDetection(num_{tp})$;
26 $\quad\quad$ **if** $Di_{tp} > thr_D$ ***or*** $AVG_{cov} > avgSpeedRegion$ ***or*** $r == r_{max}$ **then**
27 $\quad\quad\quad \mathcal{IZC}.add(r + r_{buffer})$;
28 $\quad\quad\quad$ **break**;
29 $\quad\quad$ **else**
30 $\quad\quad\quad r \leftarrow r + r_{delta}$;
31 **return** $\mathcal{CP}$, $\mathcal{IZC}$

property of key point into two-dimensional plane data. Then we group all the key points within each cell into clusters according to the distance between projection coordinate of the points' heading, as illustrated in Fig. 5. The converting formula (1) is as follows:

$$ND = \{(x, y) | x = sin(\theta), y = cos(\theta)) \\ , \theta = p_i.direction, p_i \in Tr\} \quad (1)$$

Where ND is the set of projected coordinates of key point's direction.

It is observed that vehicles may slow down when cornering at road intersection, we further pinpoint the road intersection cells based on the average speed of neighboring cells. Thus, the cell C^l that satisfies the following two conditions can be determined as a road intersection cell (lines 11-14): *(1)*: the average speed of all the points in the cell C^l is less than that of neighboring cells; *(2)*: the number of directions (denoted

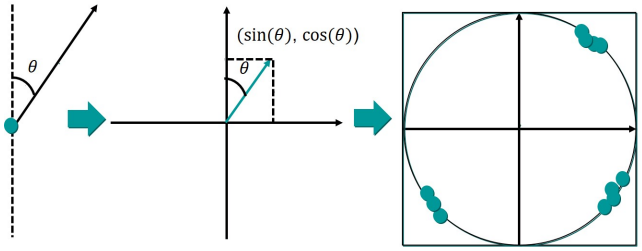


Fig. 5: An Example of *Direction-based* Clustering

as dir_{num}) of the cell C^l that obtained by direction-based clustering is greater than the minimum number threshold of directions within the cell (denoted as thr_{dmin}, thr_{dmin} is set as 2 in subsequent experiments), or dir_{num} equals to thr_{dmin} but with few differences between directions. Then, all the key points in road intersection cell are clustered using *Meanshift* method to determine the central position of that road intersection (lines 16-17). Since different sized cells may produce a few redundant central positions, e.g., the distance between any two central positions is less than 50 meters, they shall be merged (line 18).

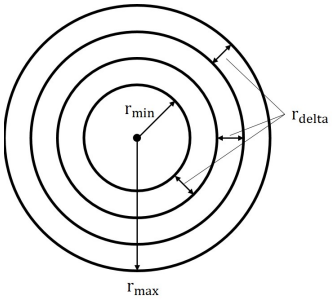


Fig. 6: Illustration of Detection for Road Intersection Coverage

2) Coverage Detection: In view of that the central position of road intersection has not always more turning behavior, e.g., *roundabout* and *overpass*, we attend to leverage annulus geometry instead of circle geometry to approximately depict the coverage of road intersection. We regard the point whose angle difference between its previous point and itself is beyond 30° as a turning point. Based on the observation that more turning points and lower speed occurs within the coverage of road intersection, we believe that with the enlargement of annulus region, the number of turning points within the annulus gradually decreases while the speed increases significantly. So we attempt to determine the coverage of road intersection by analyzing the variation of density statistics and speed for turning points passing through the annulus with constantly increasing radius (lines 19-30).

Specifically, to determine the coverage of road intersection, we treat the location of intersection as the center, and build the annulus in terms of the increasing radius, as shown in Fig. 6. In subsequent experiments, the radius increment (denoted as r_{delta}) is empirically set as 5 meters. In a bid to improve the accuracy of coverage determination, we set a minimum radius threshold (denoted as r_{min}) to avoid the situation that fewer turning points existing within the coverage, and a max radius threshold (denoted as r_{max}) to ensure the coverage not overlapping with the peripheral regions of road intersection. Here r_{min} is empirically set as 50 meters and r_{max} is set as 200 meters.

For i_{th} annulus, we calculate the average speed (denoted as AVG_{cov}), and then we compute the density (denoted as Di_{tp}) of all the turning points within that annulus by $Di_{tp} = \frac{num_{tp}}{Area_{r_i} - Area_{r_{i-1}}}$. Annulus building process will ends and the coverage of road intersection will be determined when one of the following conditions satisfies: (1) Di_{tp} is lower than the given turning density threshold thr_D; or (2)AVG_{cov} is beyond the average speed of that region; or (3) r reaches r_{max}.

C. Topology Calibration within Influence Zone

Due of time-varying traffic situation and ever-changing road infrastructure, the geometry connectivity and allowable traveling directions of road intersection are required to be timely updated. To that end, we try to find the topological changes within road intersection influence zone and using them to calibrate the road map. As described in Algorithm 3, first, we derive the trajectories within influence zone in terms of the central position of the intersection (denoted as $\mathcal{CP}$) and influence zone coverage (denoted as IZC) (line 3). Then we employ *DBSCAN* method based on $\boldsymbol{Frechét}$ distance to obtain various clusters that represent different heading directions, e.g, *go straight*, *turn right* and *turn left*, etc (lines 4-8). In order to derive the centerline for each turning trajectory cluster, we design a three-step fitting strategy, which is comprised of $\boldsymbol{Frechét}$-based sampling, *force attraction*-based clustering and *B-spline* smoothing method (lines 9-12). Finally, by matching the generated centerlines with the existing map, we try to figure out the topological changes within road intersection influence zone. They mainly involve missing or incorrect turning behaviors due of new built (blocked or re-opened road), missing of hooking relation between roadways and turning path deviation, etc.

1) Turning Cluster Extraction: To obtain different orientations within road intersection influence zone, our clustering task is to divide the set of trajectory segments that derived by connecting any two adjacent key points within road intersection influence zone into clusters in terms of specified similarity measurement.

Similarity Measurement. Since $\boldsymbol{Frechét}$ distance [24] is widely applied to measure the similarity between two curves, we leverage $\boldsymbol{Frechét}$ distance to assess the similarity of trajectory segments, which can be calculated as below:

$$f(Tr_a, Tr_b) = d_{Frechét}(Tr_a, Tr_b) \tag{2}$$

Its corresponding matrix can be constructed by Equation (3).

$$M_{Frechét} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix} \tag{3}$$

Consider that complex road intersections are of different size, a unified similarity distance threshold is not applicable to judge whether the trajectories are similar when clustering. Therefore, we use an adaptive similarity threshold instead of a unified one. $\boldsymbol{Frechét}$ distance between the trajectories is first calculated by Equation (2), and the similarity threshold ζ is determined by $\zeta = \frac{median(M_{Frechét})}{\beta}$, i.e, the median of $\boldsymbol{Frechét}$ distance matrix divides by β. In subsequent experiments, β is empirically set as 3.5.

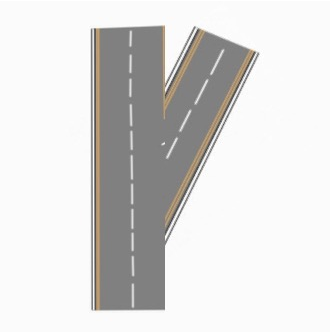


Fig. 7: A Special Example

However, $\boldsymbol{Frechét}$ distance is hard to distinguish different trajectories especially for more complicated road topology. As shown in Fig. 7, the trajectories from two neighboring roadways but with a little heading direction difference cannot be identified by $\boldsymbol{Frechét}$ distance. To address this issue, we incorporate direction weight into similarity measurement. After we use Equation (4) and (5) to construct a direction similarity weight matrix between any two trajectories (denoted as Tr_1 and Tr_2), we calculate angle gaps between the starting points and ending points of Tr_1 and Tr_2 respectively. If the angle gap of starting points (or ending points) is greater than $15°$, the weight is set to 1, otherwise 0.

$$w(Tr_a, Tr_b) = \begin{cases} 1, & if\ dirDiff(Tr_a, Tr_b) \geq \gamma \\ 0, & otherwise. \end{cases} \tag{4}$$

$$W_{direction} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \tag{5}$$

Finally, the similarity between any two trajectories is obtained by the following Equation (6) (lines 4-7).

$$TSM = M_{Frechét} + W_{direction} * \zeta \tag{6}$$

***DBSCAN* Clustering.** Based on the result of similarity measurement (Equation (7)) between trajectories, we leverage *DBSCAN* method to group trajectories within influence zone into *turning clusters* that represent different heading directions (line 8). For two important parameters of *DBSCAN* method, i.e, *minSample* and *eps*, we empirically set *minSample* as 3, which is used to ensure each cluster has at least 3 trajectories, and meanwhile use ζ to determine the value of *eps*.

$$Similar(Tr_i, Tr_j) = \begin{cases} true, & if\ TSM[i,j] \leq \zeta \\ false, & otherwise. \end{cases} \tag{7}$$

Algorithm 3: Turning Centerline Extraction

Input: Intersections' central location set $\mathcal{CP}$, Intersection influence zone coverage set $\mathcal{IZC}$, The set of quality improved trajectories Tr;
Output: The set of turning centerlines $\mathcal{C}$;

```
1  The set of trajectories within influence zone: Tr_z ← ∅;
2  The set of turning clusters: TC ← ∅;
3  Tr_z ← getZoneTrajectories(CP, IZC, Tr);
4  TSM ← Construct trajectory similarity matrix;
5  for trajectory Tr_i in Tr_z do
6      sm ← SimilarityMeasure(Tr_i, Tr_z);
7      TSM ← TSM.update(sm);
8  TC ← DBSCAN(TSM, eps, minSamples);
9  for cluster c in TC do
10     centerLine ← fittingCenterLine(c);
11     C ← C ∪ BSplineSmoothing(centerLine);
12 return C;
```

2) Turning Centerline Fitting: In what follows, we describe the details of turning centerline fitting process. Due of the stability of *force attraction*-based clustering method [3], [11], we employ it to extract the centerlines of *turning clusters* (lines 9-10).

***Force attraction*-based method.** It first randomly samples one trajectory (denoted as Tr_i) in a cluster as a reference trajectory, and then iteratively adjusts the locations of Tr_i's points using the rest of trajectories within the same cluster. In the process of adjustment, *force attraction* method assumes that there are two forces acting on any point (denoted as p_j) of Tr_i, the new position of point p_j is obtained by searching for the position where two forces achieve a balance. The forces includes the total gravitational force (denoted as $F_1(p_j)$) of the similar trajectories to p_j, and a spring force (denoted as $F_2(p_j)$) that prevents point p_j from moving away from its original position. $F_1(p_j)$ and $F_2(p_j)$ can be calculated by the following formulas respectively.

$$F_1(p_j) = \sum -\frac{M}{\sigma^3\sqrt{2\pi}} \exp\left(-\frac{(p_j - p_k)}{2\sigma^2}\right) (d(p_j, p_k)) \cos\theta \tag{8}$$

$$F_2(p_j) = s(x - p_j) \tag{9}$$

Where p_k denotes one point of any similar trajectory, $d(p_j, p_k)$ is the shortest distance from point p_j to p_k, θ is the difference between the direction of p_k and p_j, M and σ are two parameters used to determine the potential energy of the

gravitational force (by default, $M = 1$ and $\sigma = 10$), s is a spring constant (by default, $s = 0.005$), and x is the new position of point p_j.

Frechét*-distance sampling.** For *force attraction* method, randomly sampled trajectory easily leads to the unstability of centerline fitting. Especially when randomly sampled reference trajectory is far from actual center of road, the fitting deviation is large. Therefore, we introduce the strategy of ***Frechét-distance sampling into centerline fitting process. To be specific, we randomly sample k (by default, $k = 5$) trajectories as candidate reference trajectories from each *turning cluster*, and calculate the sum of ***Frechét***-distances between each candidate and the rest trajectories of that cluster respectively. The candidate trajectory with smallest distance-sum is viewed as the reference trajectory.

***B-spline* smoothing.** In order to restore the shape of the turning path in a more realistic way, we leverage *B-Spline* [25] method to smooth the turning centerline (line 11). *B-spline* method is capable of restoring curve features of any turning centerline based on polynomial interpolations. Specifically, for any two points p_i and p_{i+1} in each turning centerline, given a knot vector $U = \{u_0, u_1, ..., u_m\}$, we utilize a cubic *B-spline* interpolation function (by default $p = 3$ and $m = 10$) to derive a *B-spline* curve between p_i and p_{i+1}, i.e. $C(u) = \sum_{i=0}^{n} N_{i,p}(u)P_i$. The final result is that the turning centerline curve is naturally fitting vehicles' traveling traces, what we called as turning path.

3) Incorrect Topology Identification: In a bid to figure out missing turning path or turning path with position offset, the extracted turning centerlines are implemented map matching with the existing map to find unmatched centerlines. We utilize *Hidden Markov-based* map matching method presented in [19], and meanwhile introduce *convex hull*-based sub-roadnetwork strategy to speed up matching process.

(a) Turning centerlines (b) Convex hull building (c) Road network joining

Fig. 8: An Illustration of Convex Hull

As illustrated in Fig. 8 (a) and (b), initially, we derive one convex hull (denoted as ch_i) to cover all turning centerlines belonging to each road intersection. Then we obtain its corresponding sub-roadnetwork through joining ch_i with the existing map, as shown in Fig. 8 (c). Hereby, we execute map matching for turning centerlines only within the sub-roadnetwork containing each road intersection instead of whole road network, which sharply upgrades the efficiency of map matching process. We set the maximum road-network region searching radius (denoted as r_{match}) of any point in the centerline by default as 25 meters. We treat each point in the turning centerline as an observed variable, and regard the corresponding sub-roadnetwork as a hidden variable for each point. Subsequently, we calculate the observation probability according to the distance between sub-roadnetwork and the midpoint of the centerline. In the meanwhile, we calculate the state transition probability based on the difference between the distance of two adjacent points and that of two matched road segments. Finally, *Viterbi* algorithm is leveraged to obtain the path with maximum probability, and through which the matching degree between turning centerline and the road network can be judged.

As one of special turning behaviors, *u-turn* often exists in the middle area of two parallel roadways with opposite heading directions. When calculating probability using a unified value of r_{match}, turning centerline of *u-turn* is easy matched with one of parallel roadways rather than additional turning path. To this end, we judge whether the extracted turning centerline belongs to *u-turn* behavior based on the angle difference between the starting point and ending point of that centerline. Specifically, we regard the case that angle difference is beyond 130° as *u-turn*, and set r_{match} for *u-turn* pattern as 11 meters (derived by multiple experimental tests). Experiments on trajectory data set demonstrates the effectiveness of this method.

V. Experimental Evaluation

In this section, we conduct extensive experimental evaluations of our proposed topology calibration method based upon trajectory data, and showcase comparative experimental results with a focus on the accuracy of topology identification for road intersection influence zone. Specifically, we assess the effectiveness of core zone identification method through comparing our proposal with previous works in terms of some evaluation criteria. More importantly, we illustrate substantial visualization comparative results to verify the stability and robustness of our topology calibration method especially for turning path generation.

Given the specific road network and vehicles' traveling traces, all parameters are established empirically by conducting a series of initial experiments and evaluating the quality of the respective results. While these results cannot write all go into detail in this paper due of the space restrict.

All codes, implemented in Python2.6, are evaluated on a 10-node clustering running Spark 2.2.0 over Ubuntu 12.0.4. Each node is equipped with an 8 cores Intel E5335 2.0 GHz processor and 16 GB memory.

A. Data set

For our evaluation, we utilize two real trajectory data sets to evaluate our proposed methods, including a publicly available shuttle bus trace data set from University of Illinois at Chicago (hereafter termed $Bus2012$) and a car's trajectory data set in *Futian* District of *Shenzhen* from *Didi Chuxing* (hereafter termed $Didi2019T$). It is worth noting that our proposal mainly depends upon the trajectory data, and requires the aid of road network data until topology calibrating phase. Therefore, the road network data set from *Didi Chuxing*

(hereafter termed $Didi2019RN$) is used to assist in finding topological changes within influence zone.

$Bus2012$ is a collection of 889 shuttle bus trajectory data from the University of Illinois campus shuttles in Chicago. It contains the attributes of Timestamp, Latitude as well as longitude coordinates and Angle, and has the sampling rate of 3 to 4 seconds. In a bid to reduce the imprecision of measured angle of sampling points and at the same time supplement speed information, we use Longitude and Latitude coordinates of consecutive sampling points to infer heading direction and speed for each point.

$Didi2019T$ is a trajectory data set generated by thousands of cars from *Didi Chuxing* every day. It contains the attributes of Vehicle ID, Time, Longitude and Latitude, etc. and has the sampling interval of 2 to 4 seconds. Given the high-volume nature of this data set, we only select the data during peak hours for the experiments, i.e., from 9:00 am to 10:00 am on August, 2019, (similar desensitized data sets can be obtained from here †)

OpenStreetMap‡ is an editable map created by the public, which is used as the true digital road network in verifying the effectiveness of detection method on $Bus2012$.

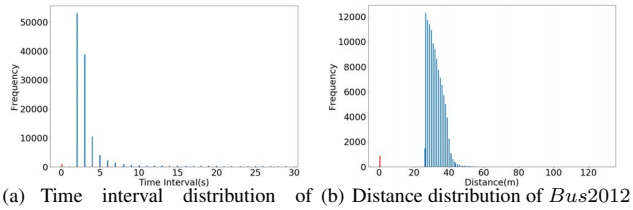
$Didi2019RN$ is a version of road network data set, the date of which is the same as our selected trajectory data set from $Didi2019T$.

Comparative Approaches. To evaluate the benefits of our proposed topology calibration method, we single out several contrast approaches, including some road intersection detection methods and map update methods.

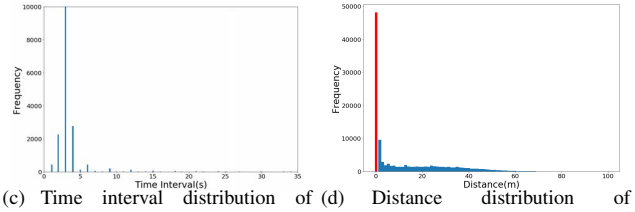
- *CBTP* [3] based on the implicit assumption that intersections exist in the region where more *turns* occurs. It attempted to pinpoint the intersections by searching high-density cells of *turns*.
- *Ahmed2012* [12] treated the endpoints of unmatched portions as the vertexes of the constructed road map through map matching.
- *Kharita* [21] tried to obtain cluster centers using k-means clustering and regards those centers as the vertexes in the road network.
- *HyMU* [20] aimed to find missing roads in the existing map through inferring road candidates for consecutive time windows and merging such candidates, and generated the road centerline of missing roads using *sweeping line* method.

Among them, we evaluate the former three methods for comparison purpose of verifying the effectiveness of our road intersection location detection method, and choose *HyMU* methods as contrast method to demonstrate the accuracy and stability of our turning path fitting strategy.

Analysis of Data. To evaluate the quality of experimental data, initially, we make the statistic analysis on the spatial distance and sampling time interval between consecutive points for two data sets. Fig. 9 illustrates the statistical results on $Bus2012$ and $Didi2019T$. One can see that for $Bus2012$, the maximum distance is 129 meters and the maximum time interval is 29 seconds, while for $Didi2019T$ the maximum distance reaches 1,170 meters and the maximum time interval is 2,178 seconds. To alleviate the effect of large distance and large interval on core zone detection of road intersection, we split raw trajectory data of two data sets into segments. In addition, as shown in Fig. 9 (b) and (d), the red bar represents the case that distance is 0, which may be caused by vehicles waiting on the light or traffic jam. To avoid the interference of such points to heading direction calculation, we attempt to remove the redundant points. As depicted in Section IV, through



(a) Time interval distribution of $Bus2012$ (b) Distance distribution of $Bus2012$



(c) Time interval distribution of $Didi2019T$ (d) Distance distribution of $Didi2019T$

Fig. 9: Distribution of Time Interval and Distance

improving the quality of data by segmentation, denoising and simplification, the sampling points with big distance gap and the redundant points are screened out, and then the points are reduced remarkably while keeping the turning behavior characteristic of raw trajectories. Subsequently, based upon high-quality data set, we try to detect the core zone of road intersection. This involves location detection and coverage determination for road intersection. In the end, we attend to identify the topological changes within the influence zone of road intersection, which includes turning clusters deriving and turning path generation. Only the experimental results of important phases are presented on account of space limitation.

B. Location Detection

The main task of this step is to verify the robustness and effectiveness for finding the road intersections from trajectory data. We compare $CITT$ with three comparative methods on two data sets separately, and present the quantitative results and visualization results.

1) Evaluation Criteria: To quantitatively evaluate our proposed road intersection detection method and comparative methods, we utilize *Precision*, *Recall* and *F-measure* as the evaluation criteria. The ground truth data is generated by volunteers' manual labeling according to the map data. Let L_{tru} denote the amount of the ground truth road intersections, L_{det} denote the total number of the identified road intersections by

†https://gaia.didichuxing.com
‡https://www.openstreetmap.org/

the detection method, and L_{corr} denote the amount of the correctly identified road intersections. The higher the *F-measure* is, the better the detection method performs. *Precision*, *Recall* and *F-measure* are defined respectively as below:

$$Precision = \frac{L_{corr}}{L_{det}}, \quad Recall = \frac{L_{corr}}{L_{tru}},$$
$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}.$$

TABLE I: Quantitative Results on $Bus2012$

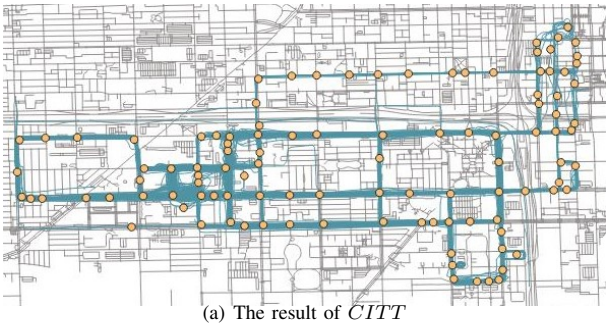
Detection Method	Evaluation Criteria		
	Precision	Recall	F-measure
CITT	**0.60256**	**0.88679**	**0.71756**
$CBTP$	0.43284	0.54717	0.48333
$Ahmed2012$	0.25	0.86792	0.38819
$Kharita$	0.23767	1	0.38406

2) Quantitative Comparison: As shown in TableI, $CITT$ can achieve a significantly higher $Precision$ and $F - measure$ score than the other methods, which demonstrates superior performance of $CITT$ for detecting the locations of road intersection. This is due to that $CITT$ has better robustnessfor finding the road intersections of different sizes by using top-down *quad tree*-based cell division method. Additionally, $CITT$ tries to figure out road intersection cells by implementing *direction-based clustering* and *speed-based checking strategy*. While $Kharita$ achieves good recall score on *Bus2012*, the reason is that the vertexes found by it contain all the road intersections and some possible breakpoints on the roadways.

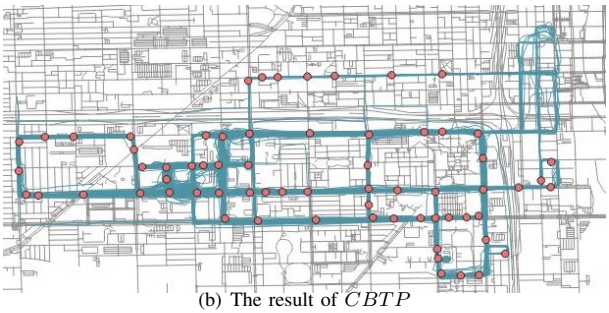
3) Visualized Comparative Results: We conduct some visual comparison evaluations on $Bus2012$ and $Didi2019T$ to reveal the quality distinctions between the location detection results by $CITT$ and $CBTP$ method.

Analysis on $Bus2012$: First, we give visualized comparative results of $CITT$ and $CBTP$ method on $Bus2012$, as shown in Fig. 10. The identified locations of road intersections are depicted in yellow circles (for $CITT$) and red circles (for $CBTP$) respectively, which overlay on top of the corresponding trajectories in road network (marked with light blue lines). As can be seen from Fig. 10 (a) and (b), although both of the approaches inevitably have misjudgment to some extent, e.g. several wrong detected intersections on the highway, our proposal can find more road intersections than $CBTP$, and the detection result is visually close to the actual road network. It is also consistent with quantitative comparison results on $Bus2012$.

Analysis on $Didi2019T$: Subsequently, we compare $CITT$ with $CBTP$ on a selected test region from *Futian* district of *Shenzhen* using $Didi2019T$, as is magnified in Fig. 11. As compared to $Bus2012$, this test region is larger and has more complex topological. The trajectories are depicted in deep blue lines and the detected locations of road intersections are highlighted in red circles. From the experimental results shown in Fig. 13 (a) and (b), we can find that 124 road intersections



(a) The result of $CITT$



(b) The result of $CBTP$

Fig. 10: Results of Location Detection on $Bus2012$

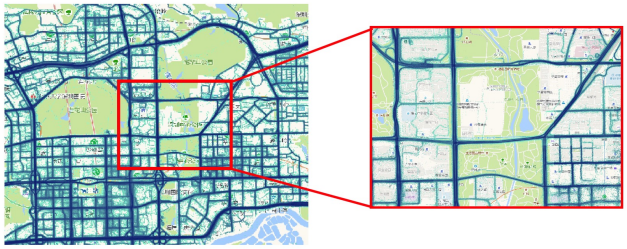


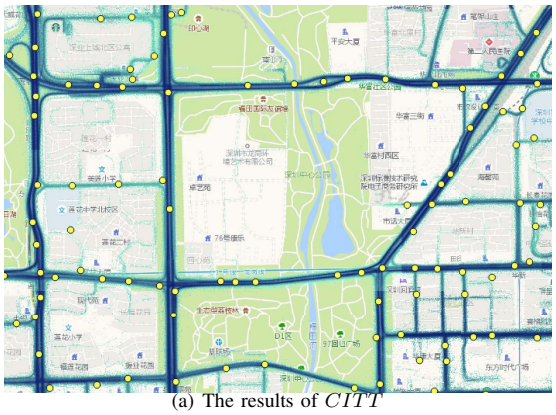
Fig. 11: A Testing Region from $Didi2019T$

are found by $CITT$, while $CBTP$ only pinpoints 59 road intersections. Obviously, our proposal obtains a more stable detection result than $CBTP$.

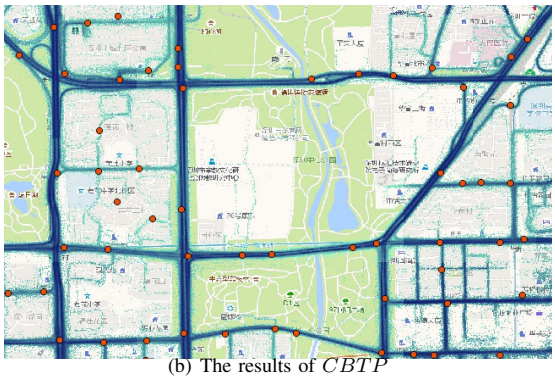
The goal of our calibration method is to find the changing turning paths within the influence zone of road intersection, and update the existing map with them. According to the detected location and coverage of road intersection, we obtain the influence zone and then extract the trajectories within it. On the basis of that, we attempt to group these trajectories into various clusters in terms of heading directions.

C. Turning Cluster Deriving

We implement $CITT$ on $Didi2019T$ to assess the performance of our turning cluster deriving process. Through multiple experiments, it is validated that our proposal can distinguish turning clusters of various complex intersections. As illustrated in Fig. 13, it can be seen that in the vicinity of a complex intersection, there are multiple turning paths and straight paths. The trajectories within this road intersection influence zone are mixing together, which are difficult to differentiate. By executing *DBSCAN* algorithm based on



(a) The results of $CITT$



(b) The results of $CBTP$

Fig. 12: Visualized Detection Result on $Didi2019T$

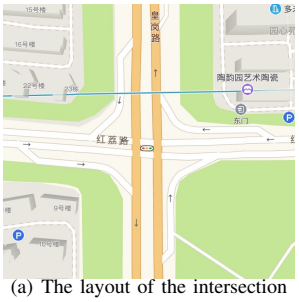
Frechét distance, total 12 clusters are derived (highlighted in different color lines). They include 8 turning paths and 4 straight paths, each of which contains a certain number of trajectories with the same heading direction.

D. Turning Centerline Fitting

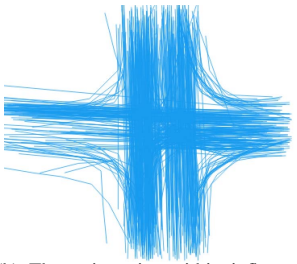
To obtain smooth turning path, we adopt a three-step strategy consisting of $Frechét$-based sampling, *force-attraction* adaption and *B-Spline* smoothing. We conduct comparative experiments with $HyMU$ to verify the stability of our proposed centerline fitting method. The visualized comparative results is shown in Fig. 14. $HyMU$ employs *sweeping line* method to infer the centerline of missing road. One can see that the deviations between our generated turning centerline and referenced road data are obviously smaller than $HyMU$. Whether the case of various turning centerlines mixing together (as illustrated in Fig. 14 (a) and (b))or the case of special turning centerline like U turn (as illustrated in Fig. 14 (c) and (d)), $CITT$ outperforms $HyMU$.

E. Topology Calibration

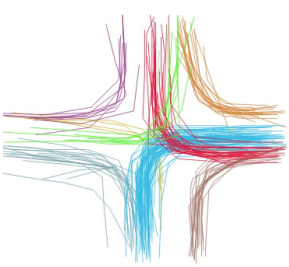
Since the extracted turning paths are implemented map matching with the existing map to find unmatched paths, the calibration of turning path within road intersection influence zone requires the support of corresponding road network data $Didi2019RN$. In order to evaluate the effectiveness of our solution, we focus on updating two types (i.e. missing turning



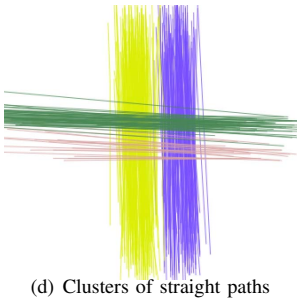
(a) The layout of the intersection



(b) The trajectories within influence zone

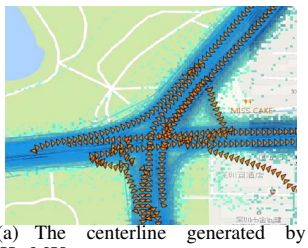


(c) Clusters of turning paths

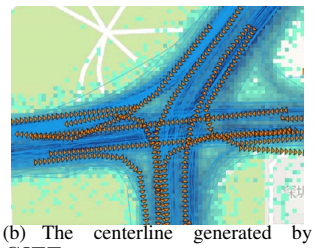


(d) Clusters of straight paths

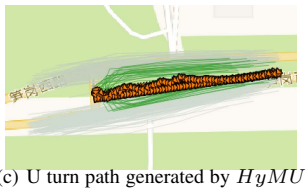
Fig. 13: Turning Clusters Deriving



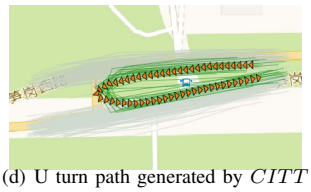
(a) The centerline generated by $HyMU$



(b) The centerline generated by $CITT$



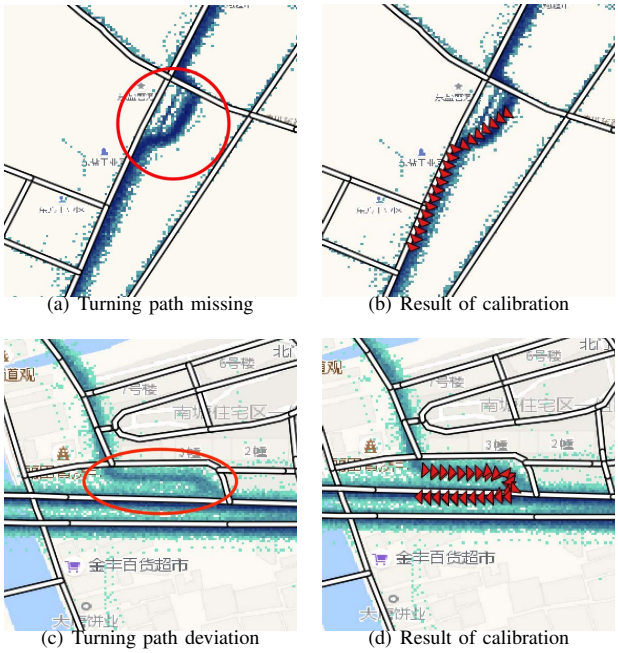
(c) U turn path generated by $HyMU$



(d) U turn path generated by $CITT$

Fig. 14: Turning Clusters Fitting

path and turning path with position offset) of incorrect topologies on $Didi2019T$ and present the visualized experimental results, as shown in Fig. 15. The trajectories are depicted in deep blue lines, the missing or incorrect turning paths are marked with big red circles, and corrected turning paths are highlighted in red indented lines. From the results shown in Fig. 15(b) and 15(d), we can observe that our proposal effectively detect the missing turning paths and incorrect paths within road intersection influence zone. In the experiments of $Didi2019T$, we detect 101 road intersections in total. Then we conduct the result analysis by combining with the road network data and trajectory heat-map. For the detected results, there are 69 correct ones, among which 58 road intersections exist turning path deviation and 11 road intersections have



(a) Turning path missing (b) Result of calibration

(c) Turning path deviation (d) Result of calibration

Fig. 15: Turning Path Calibration on $Didi2019T$

missing turning paths.

VI. Conclusion and Future Work

CITT offers a periodical solution based on vehicles' trajectory data to the inaccuracy problem of road intersection topology, which involves core zone detection and turning path calibration. According to the direction difference and speed variation characteristics, our proposal enables the location and coverage of road intersection to be accurately identified without road network. Further, through combining *Frechet*-based *DBSCAN* clustering with a hybrid centerline fitting strategy, turning path within the influence zone of road intersection can be effectively generated. Although the calibration procedure needs to analyze massive amounts of vehicle traces, it can be implemented on distributed platform in *Didi* lab and hence takes less time overhead. In the meanwhile, our proposal does not need the experts to edit the map, so the whole process is highly automated. In our future work, topology calibration for road intersections should realize lane-level refinement for precise navigation purpose.

Acknowledgements. The authors are very grateful to the editors and reviewers for their valuable comments and suggestions. This work is supported by NSFC (No. 61702423, U1501252).

References

[1] A. Fathi and J. Krumm, "Detecting road intersections from GPS traces," in *Geographic Information Science*, 2010, pp. 56–69.

[2] J. Wang, X. Rui, X. Song, X. Tan, C. Wang, and V. Raghavan, "A novel approach for generating routable road maps from vehicle GPS traces," *International Journal of Geographical Information Science*, vol. 29, no. 1, pp. 69–91, 2015.

[3] J. Wang, C. Wang, X. Song, and V. Raghavan, "Automatic intersection and traffic rule detection by mining motor-vehicle GPS trajectories," *Computers, Environment and Urban Systems*, vol. 64, pp. 19–29, 2017.

[4] L. Tang, L. Niu, X. Yang, X. Zhang, Q. Li, and S. Xiao, "Urban intersection recognition and construction based on big trace data," *Acta Geodaetica et Cartographica Sinica*, vol. 46, no. 6, pp. 770–779, 2017.

[5] X. Xie, W. Liao, H. K. Aghajan, P. Veelaert, and W. Philips, "Detecting road intersections from GPS traces using longest common subsequence algorithm," *ISPRS Int. J. Geo-Information*, vol. 6, no. 1, p. 1, 2017.

[6] X. Xie and W. Philips, "Road intersection detection through finding common sub-tracks between pairwise GNSS traces," *ISPRS Int. J. Geo-Information*, vol. 6, no. 10, p. 311, 2017.

[7] L. Li, D. Li, X. Xing, F. Yang, W. Rong, and H. Zhu, "Extraction of road intersections from GPS traces based on the dominant orientations of roads," *ISPRS Int. J. Geo-Information*, vol. 6, no. 12, p. 403, 2017.

[8] M. Pu, J. Mao, Y. Du, Y. Shen, and C. Jin, "Road intersection detection based on direction ratio statistics analysis," in *MDM*, 2019, pp. 288–297.

[9] M. Deng, J. Huang, Y. Zhang, H. Liu, L. Tang, J. Tang, and X. Yang, "Generating urban road intersection models from low-frequency GPS trajectory data," *International Journal of Geographical Information Science*, vol. 32, no. 12, pp. 2337–2361, 2018.

[10] Y. Huang, Z. Xiao, X. Yu, D. Wang, V. Havyarimana, and J. Bai, "Road network construction with complex intersections based on sparsely sampled private car trajectory data," *TKDD*, vol. 13, no. 3, pp. 35:1–35:28, 2019.

[11] L. Cao and J. Krumm, "From GPS traces to a routable road map," in *GIS*, 2009, pp. 3–12.

[12] M. Ahmed and C. Wenk, "Constructing street networks from GPS trajectories," in *ESA*, 2012, pp. 60–71.

[13] S. Karagiorgou and D. Pfoser, "On vehicle tracking data-based road network generation," in *SIGSPATIAL*, 2012, pp. 89–98.

[14] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse GPS traces for map inference: comparison of approaches," in *KDD*, 2012, pp. 669–677.

[15] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu, "Crowdatlas: self-updating maps for cloud and personal use," in *MobiSys*, 2013, pp. 27–40.

[16] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk, "A comparison and evaluation of map construction algorithms using vehicle tracking data," *GeoInformatica*, vol. 19, no. 3, pp. 601–632, 2015.

[17] H. Wu, C. Tu, W. Sun, B. Zheng, H. Su, and W. Wang, "GLUE: a parameter-tuning-free map updating system," in *CIKM*, 2015, pp. 683–692.

[18] H. Li, L. Kulik, and K. Ramamohanarao, "Automatic generation and validation of road maps from GPS trajectory data sets," in *CIKM*, 2016, pp. 1523–1532.

[19] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. J. Guibas, "City-scale map creation and updating using GPS collections," in *SIGKDD*, 2016, pp. 1465–1474.

[20] T. Wang, J. Mao, and C. Jin, "Hymu: A hybrid map updating framework," in *DASFAA*, 2017, pp. 19–33.

[21] R. Stanojevic, S. Abbar, S. Thirumuruganathan, S. Chawla, F. Filali, and A. Aleimat, "Robust road map inference through network alignment of trajectories," in *SDM*, 2018, pp. 135–143.

[22] J. Wu, Y. Zhu, T. Ku, and L. Wang, "Detecting road intersections from coarse-gained GPS traces based on clustering," *JCP*, vol. 8, no. 11, pp. 2959–2965, 2013.

[23] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.

[24] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti Del Circolo Matematico Di Palermo*, vol. 22, no. 1, pp. 1–72, 1906.

[25] Y. Huang, Z. Xiao, X. Yu, D. Wang, V. Havyarimana, and J. Bai, "Fast hierarchical clustering routines for r and python," *Journal of Statistical Software*, vol. 53, no. 9, pp. 1–18, 2017.