Trajectory Inference for Mobile Devices Using Connected Cell Towers

Kushani Perera, Tanusri Bhattacharya, Lars Kulik, James Bailey
Department of Computing and Information Systems
The University of Melbourne, Victoria 3010, Australia
{bperera, tanusrib}@student.unimelb.edu.au,{lkulik, baileyj}@unimelb.edu.au

ABSTRACT

Trajectory inference from raw location samples of a mobile device is an important task for many location based services, such as crowd sourced traffic monitoring, fleet management and personalized trip planning. This task becomes challenging when location samples are obtained only from the connected cell towers (GSM localization), instead of using other localization sensors such as GPS or Wi-Fi. Cell tower based localization consumes negligible energy compared to GPS or Wi-Fi and has high availability. However, it can have large inaccuracy, making the task of cellular trajectory mapping extremely challenging. In previous studies, cellular trajectory inference has been performed assuming the availability of knowledge of the cellular network or the signal strengths of the neighbouring cell towers. However, for a mobile application running on a user's device, this information may be hard to obtain and it may also require additional storage and computation costs. In this paper, we propose a novel cellular trajectory inference method which requires only the user's connected cell tower location, time and speed information. Exploiting the preciseness of the time dimension, we accurately compute the distance a user has travelled within a cell and use it to infer the straight line segments and turning points of a trajectory. We show that using the distance information of three consecutive cells, exact inference of the line segment is possible. Our method achieves high accuracy for trajectory inference in urban areas with high cell density and straight line road segments. It does not require any historical trajectory information or pre-training and incurs low storage and computation costs.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Performance

SIGSPATIAL '15, November 03-06, 2015, Bellevue, WA, USA
 ISBN 978-1-4503-3967-4/15/11$15.00
DOI: http://dx.doi.org/10.1145/2820783.2820804.

Keywords

Cellular trajectory inference, mobile computing, cell towers, partial spatial knowledge

1. INTRODUCTION

With the ubiquitous usage of mobile phones, exploiting them as location sensing devices has emerged as an attractive concept [17]. Mobile phone based location sensing is abundantly used in many applications, such as providing location based services, network traffic studies, traffic flow analysis, fleet management, people and object tracking applications. One important task in these applications is to infer user trajectories from the location samples obtained from mobile devices.

For mobile phones, GPS has been the most popular technology for trajectory data capture in outdoor environments[1]. However, GPS based location acquisition has limited applicability in certain contexts due to high power consumption and poor coverage. Receiving continuous GPS signals causes the rapid depletion of the phone battery, restricting long timespan tracking [10, 17, 18]. GPS signals may not be available in covered areas such as indoor environments and areas under thick foliage [12, 17].

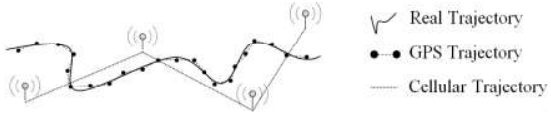
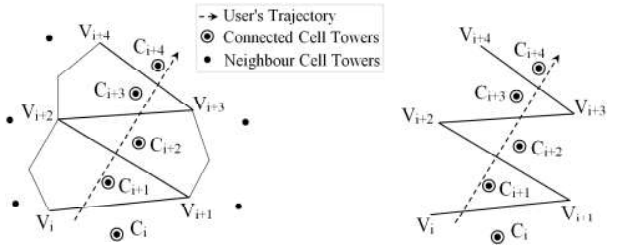


Figure 1: Imprecision in cellular trajectories due to coarse and infrequent location observations

Compared to GPS, cellular network radio signals are advantageous due to low energy consumption and higher availability [10]. Therefore, cellular network positioning is a good alternative [12, 17, 18]. However, the challenge for cellular network positioning is the low precision associated with cellular trajectories [6, 19]. In cellular network positioning, the sequence of cell tower locations a user's phone is connected is used as an approximation of the actual trajectory and is referred to as the cellular trajectory. At a given time, the user's actual location maybe anywhere within the coverage area of the connected cell tower, which may be ~1 km^2. This is a very coarse location estimation. Furthermore, the distance between two cell tower locations is ~1 km on average, therefore a user's location change is observed only

[1]https://www.glympse.com, http://www.google.com/mobile/mytracks



(a) Using full knowledge about the neighbour cell tower distribution

(b) Using only the connected cell tower location information

Figure 2: Voronoi cell computation in trajectory inference

infrequently. The user's trajectory behaviour is thus uncertain and the trajectory obtained by connecting the sample points is imprecise. Figure 1 shows the high imprecision of a cellular trajectory compared to a GPS trajectory.

Most existing cellular trajectory inference methods use knowledge of the cellular network infrastructure, which is available only to the mobile network operators [3, 6, 12, 19, 20, 21, 22, 23]. For example, in a Voronoi based trajectory inference [6, 19, 21, 23], the cellular network is modelled as a Voronoi diagram and the user's trajectory as a Voronoi cell sequence. The computation of the cell boundaries requires full knowledge about the spatial distribution of cell towers in the network [2, 7, 19]. Yet, for a locally running mobile application, only the connected cell tower location sequence (partial knowledge) and cell handover times can be obtained [4,11,16]. Furthermore, the Voronoi based methods produce an imprecise trajectory (cell area ~1 km^2) and incur high computation and memory costs [7, 8] and are thus less appropriate for applications running on mobile platforms. Signal strength based methods [1, 4, 5, 13, 15, 18, 22] also require knowledge such as the spatial distribution of neighbouring cell towers and their signal strengths. Furthermore, they are susceptible to various problems such as the inaccuracy of signal strength measurements, device dependency, high storage and computation costs, and long term training requirements [4, 15, 18].

In this paper, we propose a device centric cellular trajectory inference method using partial knowledge about the cellular network. It requires only the user's connected cell tower locations, cell handover times and speed information which are either locally accessible to (or can be computed in) a mobile device. For example, a user's speed information can be estimated using the phone accelerometer and gyroscope[2]. As shown in Figure 2b, our method computes the partial Voronoi cell sequence for a trajectory from the connected cell tower locations. As a mobile phone user's cell transition usually occurs to an adjacent cell, his/her trajectory is a sequence of adjacent Voronoi cells. Therefore, for each cell, its two adjacent cells in the sequence are its Voronoi neighbours, whose cell tower locations are available for the user (as the user has connected to these cell towers). A Voronoi edge can be computed for each neighbour, resulting in two edges for a given cell (E.g.: V_iV_{i+1} and $V_{i+1}V_{i+2}$ for C_{i+1}, in Figure 2b). Compared to the Voronoi cell sequence, computed with full knowledge about the cell tower distribution (Figure 2a), this partial Voronoi cell sequence has produced a highly imprecise trajectory.

Using cell handover time and user speed information, we compute the distance the user travelled within each cell and use it to infer the straight line segments of the trajectory. We show that if the user travels along a straight line path for three consecutive cells, using distance information of the three cells, exact inference of the straight line is possible. Therefore, our method achieves high accuracy for trajectories which are combinations of line segments of longer length. Such trajectory patterns are common in urban areas, where the road networks consist of straight line segments and the cell density is high. The accuracy achieved for such patterns is significant even though not as high as GPS sometimes.

Furthermore, our method incurs low storage (no location history is stored) and computation costs (time complexity is linear to the number of connected cell towers), therefore is suitable for mobile platforms. As no location history and pre-training is required, it is even applicable for unfamiliar environments. The main contributions of our method are:

1. It requires only partial knowledge of the cellular network.
2. It can infer long line segments and turning points of a trajectory with high precision.
3. It has low storage and computation costs and is therefore well suited to mobile platforms.
4. It can be deployed for unfamiliar environments and no pre-training is required.

2. RELATED WORK

Voronoi methods are widely used in system centric cellular trajectory inference approaches. They require full knowledge of the cellular network, and are common where only a rough trajectory estimation is satisfactory [6, 19, 21]. They are inapplicable for device centric inference approaches and where precise trajectory estimations are required. Moreover, complete Voronoi diagram construction in these approaches [2, 7] incurs unnecessary computation and memory costs. Single Voronoi cell computation methods [14, 23] have reduced this memory requirement by computing only the cells traversed by the user. However, achieving time efficiency in computation still remains a challenging task.

Table 1: Applicability of cellular trajectory inference methods in a device centric setting

Inference Method	Neighbour Cell Tower Information	Pre-training	Device Dependency	Memory & Storage Costs
Voronoi	✓	✗	✗	High
Triangulation	✓	✗	✓	Low
MonteCarlo	✓	✓	✓	High
Fingerprinting	✗	✓	✓	High
TPDA	✗	✗	✗	**Low**

Cell tower triangulation is another commonly used handset positioning method which uses Received Signal Strength Indicator (RSSI) at a given position as a distance measure. However, RSSI causes high inaccuracy and ambiguity in location estimations [3, 9]; consequently, most cellular triangulation approaches have reported ~150 m error [3, 5, 15, 20]. They also require knowledge about the spatial distributions of neighbouring cell tower locations and their signal strength information, which are not guaranteed

[2]http://www.chipworks.com/about-chipworks/overview/blog/inside-the-iphone-5s, http://www.addictivetips.com/ios/go-pedometer-view-walk-speed-calories-burned-without-gps-iphone

to be available. Monte Carlo methods [1, 13, 22] increase the accuracy of cellular trajectory inference by predicting a user's location where required. However, they are computationally intensive and require data collection at high sample rates to gain high accuracy, therefore not suitable for mobile platforms. Fingerprinting [4, 15, 18] requires no knowledge about cellular network infrastructure and has achieved high accuracy, yet it requires long term training, therefore cannot be used to track users in unfamiliar environments. It is also problematic for mobile platforms due to high storage costs and is highly susceptible to device dependency problem [4]. Thus none of these inference methods is well suited for a device centric setting. To address this, we propose a device centric cellular trajectory inference algorithm, Trajectory Pattern Detection Algorithm (TPDA), whose strengths against existing methods are summarised in Table 1.

3. PROPOSED METHOD

We propose a Voronoi based cellular trajectory inference method. It uses only device centric knowledge, which is easily accessible by a mobile phone device such as the connected cell tower location sequence, corresponding handover times, and the user's speed. In this method, we suggest exploiting the preciseness of the time dimension of a cellular trajectory. Accurate time measurements allow precise computation of cell handover times, when collected at high sample rates. This, along with the user's speed information is used to compute the distance a user has travelled within each cell. Using the distance information of three or more consecutive cells, we accurately infer the straight line trajectory segments which cover these cells and approximate other trajectory segments accordingly.

3.1 Preliminary Concepts

In this section, we describe some preliminary concepts used in our algorithms. Our technique is based on the assumption that a user connects to the geographically nearest cell tower. With full knowledge of cell tower distributions, we can model the cellular network as a Voronoi diagram and the trajectory cells traversed by the user as an adjacent Voronoi polygon sequence as shown in Figure 2a. However, with partial knowledge of connected cell tower locations$(C_i, C_{i+1}, C_{i+2}, \dots)$, we compute the Voronoi edges as the perpendicular bisectors between each adjacent cell pair. For example, in Figure 2b, $V_iV_{i+1}, V_{i+1}V_{i+2}, V_{i+2}V_{i+3}, \dots$ are the computed Voronoi edges. This produces a partial Voronoi cell sequence and the user's actual trajectory is a 2-D poly line or curve intersecting this cell sequence.

Cell Cut Length (CCL): The intersection points of a user's trajectory with the Voronoi polygon boundaries (or edges) represent the entry and exit points to and from the cell. The poly line length between these intersection points is referred to as the *Cell Cut Length* of the trajectory within that cell. Figure 3 shows CCLs for three different trajectories. The trajectory's CCL within a cell is equal to the distance d, the user has travelled within that cell. This can be computed as d=(T_{exit} - T_{entry})* v, where v is the average user speed and T_{exit} - T_{entry} is the time, the user has spent within the cell. T_{entry} and T_{exit} are the cell handover times, i.e., the times when the user has crossed a cell boundary to enter into and exit from the cell, respectively. The handover

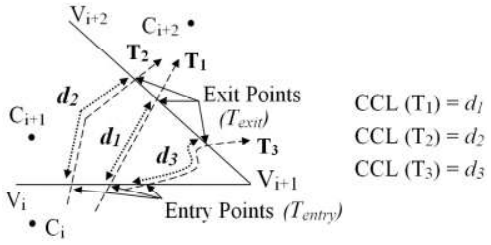


Figure 3: Cell cut lengths of different shaped trajectories within same cell boundaries

times can be precisely computed and we assume the user's speed information is available.

Definition 1. The CCL of the trajectory within a cell is the length of the trajectory between the entry and exit points of the trajectory, to and from the cell.

Number of Cell Cuts (NCC): Given a line segment, the *Number of Cell Cuts* is the number of adjacent cells whose two edges are intersected by the line segment. Figure 2b shows a line segment with three cell cuts. For a given cell density, the NCC of a trajectory line segment is an approximation for its geographical length.

Definition 2. Given a line segment (l) with the start and end points, S and E, and l intersecting all the cells in an adjacent cell sequence $C_i : C_j$, such that $S \in C_i$ or $S \in e_{i,i+1}$ and $E \in C_j$ or $E \in e_{j-1,j}$, the NCC of $l = |C_i : C_j| - 2$, where $e_{x,y}$ is the boundary between C_x and C_y, and $|.|$ is the cardinality of a cell sequence.

k-Cell Cut: A *k-cell cut* is a line segment, with a k number of cell cuts. This shows the straight movement of the user, without any direction change. Figure 2b shows a 3-cell cut.

Definition 3. A k-cell cut is a line segment whose NCC=k, where $k \in \mathbb{Z}^*$, $\mathbb{Z}^* = \mathbb{Z}^+ \cup \{0\}$.

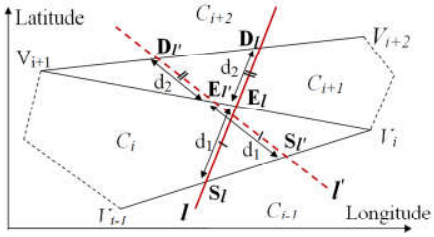


Figure 4: Traversal equivalence class for C_i:C_{i+1}; straight lines intersecting the same edge pairs with the same CCLs

Traversal Equivalence for a Single Cell: A straight line segment of a trajectory represents a straight line (L) on the 2-D plane, with a unique slope (m) and intercept (c). The set of possible straight lines on the 2-D plane is L ={(m, c), $m \in (-\infty, \infty)$; $c \in (-\infty, \infty)$ }, representing all possible straight line paths a user can take. Given a Voronoi polygon, a straight line can intersect at most two polygon boundaries. There also exist more than one line which intersect the same polygon edge pair with the same CCL, but at different intersection points. Using this concept, we define the equivalence relation on set L, with respect to a given

Voronoi cell, as the line segments intersecting the same cell edge pair with the same CCL. We name this as *Traversal Equivalence Relation (TER)* for a given cell C ($\sim_C$).

Based on the TER, set L can be partitioned into a set of equivalence classes, each class representing a different cell edge pair and CCL combination. All the lines which have the same CCL within the same cell boundary pairs belong to the same equivalence class. We name such an equivalence class as a *Traversal Equivalence Class (TEC)* for a given cell C ($[l]_C$). For example, as shown in Figure 4, l, $l^{'} \in [l]_{C_i}$.

Definition 4. Assume a line segment l that overlaps three adjacent cells C_{i-1}, C_i, C_{i+1} of a Voronoi diagram. Let L denote the set of all lines on the 2-D plane. If $V_{i-1}V_i$ and V_iV_{i+1} are edges of C_i such that $\exists S_l, E_l \in l \wedge S_l \in V_{i-1}V_i \wedge E_l \in V_iV_{i+1}$, then we define the TEC of l with respect to C_i as: $[l]_{C_i} = \{l^{'} \in L | l \sim_{C_i} l^{'}\}$

$l \sim_{C_i} l^{'} \Leftrightarrow \exists S_{l^{'}}, E_{l^{'}} \in l^{'} \wedge S_{l^{'}} \in V_{i-1}V_i \wedge E_{l^{'}} \in V_iV_{i+1} \wedge |S_lE_l| = |S_{l^{'}}E_{l^{'}}|$

Traversal Equivalence for a Cell Sequence: For the scenarios, a straight line intersects an adjacent cell sequence, $C_i : C_j$, a TER $\sim_{C_i:C_j}$ can be defined with respect to $C_i : C_j$, on set L. The relation is *intersecting the same edge pair of each cell of the adjacent cell sequence, creating the same CCLs within each cell.*

In this case, an equivalence class ($[l]_{C_i:C_j}$) corresponds to a different cell edge pair sequence and a CCL sequence. All the lines which have the same CCL sequence within the same cell edge pair sequence belong to the same equivalence class. For example, in Figure 4, lines l, $l^{'} \in [l]_{C_i:C_{i+1}}$.

In this scenario, different TERs can also be defined for each cell in the sequence ($\sim_{C_i}$:$\sim_{C_j}$), producing a set of TECs $[l]_{C_i} : [l]_{C_j}$. Each cell has its own set of TECs, on set L. As described by the following equation, the TEC of the entire sequence is the intersection of the TECs of single cells.

$$[l]_{C_i:C_j} = \bigcap_{k=i}^{j} [l]_{C_k} \quad (1)$$

3.2 Trajectory Inference

In this section, we describe our methodology for cellular trajectory inference. We first describe techniques to accurately infer the line segments of a trajectory. We then provide a methodology for handling the turning point segments.

3.2.1 Exact Inference of a Line Segment

As described in Section 3.1, the elements of a TEC represent the user's all possible straight line paths for a given cell edge and CCL sequence. Trajectory inference involves inferring all the elements in the corresponding equivalence class. Therefore, the precision of trajectory inference depends on the cardinality of the equivalence class. The lower the cardinality of the class the higher is the precision.

According to Equation 1, $[l]_{C_i:C_j} \subseteq [l]_{C_i}$; therefore, $|[l]_{C_i:C_j}| \leq |[l]_{C_i}|$, where $|.|$ is the cardinality of a set. As the number of cells in the trajectory cell sequence increases, the cardinality of the TEC decreases, reducing the number of possible paths. Therefore, we argue that the line segments which intersect more number of cells can be more precisely inferred than the line segments which intersect less number of cells. Our intuition is to find the minimum NCC, required to limit the cardinality of the TEC to one, so that we can uniquely identify the user's actual straight line path. Therefore, we computed the cardinality of the TEC for different k-cell cuts, incrementally increasing k, starting from one. As the cardinality of TEC for one-cell cuts amounts to infinity, we next consider the TEC for 2-cell cuts.

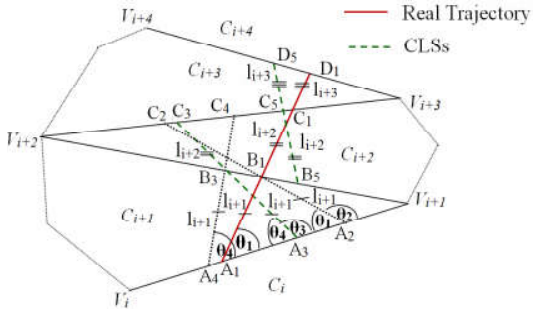


Figure 5: Candidate line segment inference for C_{i+1}:C_{i+2} and exact straight line inference for C_{i+1}:C_{i+3}

2-Cell Cut Inference: Assuming that in a cell in the cell sequence, the user's entry cell boundary is not parallel to the exit cell boundary, we show that the cardinality of TEC for 2-cell cuts is two. For details, please refer to Theorem 1 and see its proof from this link[3]. Therefore, there exist only two different 2-cell cuts which intersect the same cell edge sequence of two adjacent cells with same CCLs. Furthermore, we propose a method to exactly infer them. Figure 5 shows the 2-cell cuts, A_1C_1, A_3C_3, which have same CCLS within C_{i+1} and C_{i+2}, inferred using this method. For details about line segment and angle computations, please refer to the inference method from this link[3]. Any of these 2-cell cuts could be the user's actual trajectory, hence we refer to them as *Candidate Line Segments (CLSs)* and refer to the TEC as *Candidate Line Segments Set (CLSS).*

THEOREM 1. *Assume V_iV_{i+1}, $V_{i+1}V_{i+2}$, $V_{i+2}V_{i+3}$ are edges of two adjacent cells C_{i+1}, C_{i+2} in a Voronoi diagram. Let L denote the set of all straight lines on the 2-D plane and $|.|$ denote the cardinality of a set. Given $l \in L$, which intersects V_iV_{i+1}, $V_{i+1}V_{i+2}$, $V_{i+2}V_{i+3}$, with CCLs l_{i+1}, l_{i+2} within C_{i+1}, C_{i+2}; $|[l]_{C_{i+1}:C_{i+2}}| = |[l]_{C_{i+1}} \cap [l]_{C_{i+2}}| = 2$*

3-Cell Cut Inference: As unique identification of a straight line trajectory is still not possible, we next consider the TEC for 3-cell cuts. As shown in Figure 5, we partition the three-cell sequence (C_{i+1}:C_{i+3}) into two two-cell sequences (C_{i+1}:C_{i+2}, and C_{i+2}:C_{i+3}). For each sequence, we separately compute its CLSs (A_1C_1:A_3C_3 and B_1D_1:B_5D_5), producing two CLSSs ($[l]_{C_{i+1}:C_{i+2}}$ and $[l]_{C_{i+2}:C_{i+3}}$). If the user has travelled along the same line for three cells, there are two CLSs, A_1C_1 and B_1D_1, each from a different CLSS, which represent the same line, A_1D_1. We infer this common line as the user's trajectory within the three cells. The absence of such a common line segment means a straight line path has not occurred across the three cells. Any k-cell cut ($k > 3$) also can be uniquely identified in the same way, as they are the aggregations of adjacent 3-cell cuts.

Handling Approximate CCL Information: Sometimes only approximate CCL information is available, due to imprecise speed measurements. Therefore, when identifying the adjacent CLSs which represent the same line, we

[3]http://people.eng.unimelb.edu.au/baileyj/sigspatial_supp.pdf

compare the distance between the two intersection points of each CLS with the middle cell boundaries (C_3C_5, B_3B_5 distances for A_3C_3 and B_5D_5 in Figure 5). If both distances are below a given distance threshold, δd, the two CLSs are considered representing the same line. δd is a user adjusted parameter, based on the required precision level. For example, δd can be selected ~10 m to differentiate between two road segments.

3.2.2 *Inference of Turning Point Segments*

In the previous section, we described our methodology for accurately inferring a straight line trajectory. However, a user may change directions producing the turning points as shown in Figure 6. These can be represented as the intersections of two line segments. We refer to such trajectory segments as $k : k'$*-cell cuts*. Below we describe the scenarios of trajectory patterns with different $k : k'$*-cell cuts*.

Definition 5. A $k : k'$-cell cut is the intersection of two straight line segments (L_1, L_2) whose number of cell cuts are k and k' respectively, given that $k, k' \in \mathbb{Z}^*$, $\mathbb{Z}^* = \mathbb{Z}^+ \cup \{0\}$.

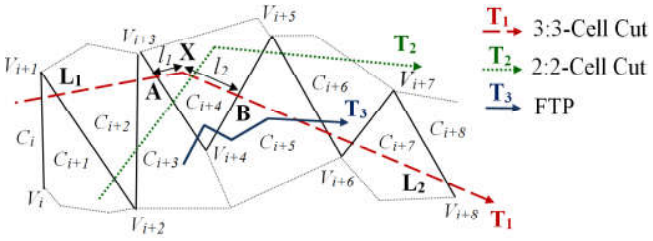


Figure 6: Different $k : k'$-cell cuts inferred by TPDA

- $\boldsymbol{k : k' \geq 3}$: Consider the trajectory T_1 in Figure 6, a 3:3-cell cut, represented as the intersection of L_1 and L_2. The line segments L_1 and L_2 can be exactly inferred using the methodology described in Section 3.2.1. However, the trajectory behaviour within the turning point cell C_{i+4} is still unknown. Algorithm 1 shows the trajectory inference method within this turning point cell. We compute the turning point as the intersection point X of L_1 and L_2. Thus, the trajectory within C_{i+4} is inferred as AXB. If X lies within the cell boundaries of C_{i+4}, next we compute the CCL of the inferred trajectory within C_{i+4} ($l_{inferred}$) as $AX+ BX$. If $l_{inferred} = l_{real}$, where l_{real} is the CCL computed using speed and handover time information; we conclude that AXB is the trajectory within C_{i+4}. Using this method, we can exactly infer the turning point and the trajectory within the turning point cell, in $k : k'$-cell cuts where $k : k' \geq 3$.
- $\boldsymbol{k, k' = 2}$: Consider the 2:2-cell cut in Figure 6 represented by the trajectory T_2. Exact inference of L_1 or L_2 is not possible as two CLSs are available for each of them, resulting in four possible line segment combinations. In such scenarios, we apply Algorithm 1 for all the combinations and select the combination whose $l_{inferred} = l_{real}$. The trajectory within the turning point cell is inferred accordingly. This shows that the exact inference of a 2-cell cut is also possible, in a 2:2-cell cut scenario.
- $\boldsymbol{k, k' < 2}$: This scenario is described by the trajectory T_3 in Figure 6. The trajectory segments are a result of frequent directional changes and are referred as *Frequent Turn Paths (FTPs)*. To infer an FTP, first we connect the end points of two adjacent straight line trajectory segments, which are already exactly inferred, with a straight line. If this line intersects all the cells, the user's mobile device connects to within the FTP, we infer this line as the user's trajectory. Otherwise, we connect the cell centres of each cell within the FTP with straight lines.

Algorithm 1: Infer the turning point of a $k : k'$-cell cut

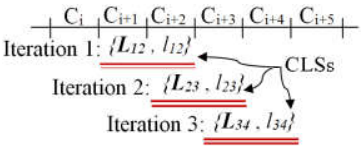
input : Line 1 (L_1), Line 2 (L_2), Cell sequence of L_1 ($\{C_i$: $C_{i+4}\}$), Cell sequence of L_2 ($\{C_{i+4}$:$C_{i+8}\}$), CCL within C_{i+4} (l_{real})
output: Intersect point of L_1 and L_2 ($turnPoint$)

```
[V_{i+3}V_{i+4}, V_{i+4}V_{i+5}] ← Perpendicular bisectors between C_{i+3}-C_{i+4} and C_{i+4}-C_{i+5};
X ← Intersect point of L_1 and L_2;
A ← Intersect point of L_1 and V_{i+3}V_{i+4};
B ← Intersect point of L_2 and V_{i+4}V_{i+5};
l_inferred ← l_1 + l_2 ← |AX| + |BX|;
if X ∈ C_{i+4} ∧ l_inferred = l_real then
    turnPoint ← X;
    else
        turnPoint ← null ;
    end
end
return turnPoint;
```

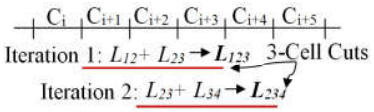
3.3 Trajectory Pattern Detection Algorithm

In the previous sections, we discussed the inference methods for different trajectory patterns. In this section, we propose our Trajectory Pattern Detection Algorithm (TPDA) for efficient application of these methods to infer a complete trajectory, which is a combination of these patterns. TPDA consists of following six steps. Given a cell sequence, traversed by a user, at each step, TPDA finds a different trajectory pattern, iteratively applying the corresponding inference method to adjacent cells in the cell sequence. The worst case time complexity of each step is linear to the total number of cells in the trajectory (n), therefore the overall time complexity of the algorithm is linear to n.

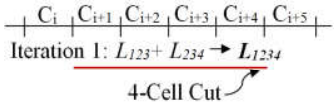
1. **Infer CLSs:** As shown in Figure 7a, Step 1 infers CLSs for each adjacent cell pair, applying the 2-cell cut inference method, discussed in Section 3.2.1, in the sliding window method.
2. **Infer 3-cell cuts:** As shown in Figure 7b, Step 2 infers 3-cell cuts occurred in each adjacent three cell sequence, applying the 3-cell cut inference method, discussed in Section 3.2.1, in the sliding window method. Adjacent CLSs, representing a common line, are aggregated as 3-cell cuts, filtering out the inappropriate CLSs.
3. **Infer k-cell cuts, $k > 3$:** As shown in Figure 7c, Step 3 infers k-cell cuts ($k > 3$), aggregating the adjacent 3-cell cut pairs which share two common cells and represent the same line. Each consecutive 3-cell cut pair, inferred in Step 2, are considered in the sliding window method.
4. **Infer 2:2-cell cuts:** After Step 3, the inferred trajectory may contain gap cells, in which no line segments are detected. Within each gap cell sequence, Step 4 searches and infers 2:2- cell cuts, iteratively applying the 2:2-cell



(a) CLS inference for adjacent cell pairs



(b) 3-cell cut inference, aggregating adjacent CLSs



(c) k-Cell Cut ($k > 3$) inference, aggregating adjacent 3-Cell Cuts

Figure 7: Sliding window method for k-cell cut ($k \geq 3$) detection in a cell sequence

cut inference method, discussed in Section 3.2.2.

5. **Infer $k : k'$-cell cuts:** Applying Algorithm 1 for each consecutive line segment intersection, Step 5 searches for $k : k'$-cell cuts ($k, k' \geq 2$) occurred and infers the trajectory within the turning point cells.
6. **Infer FTPs:** For each gap cell sequence, remaining after Step 5, Step 6 applies the $k : k'$-cell cut ($k, k' < 2$) inference method, discussed in Section 3.2.2.

3.4 Removing False Line Segments

In the CCL based line inference, discussed in Section 3.2.1, a 3-cell cut detection is possible even for a turning point trajectory segment, if the turning point segment also has the same CCL sequence as the 3-cell cut. Figure 8 shows such a scenario. The turning point path, ADE, is the user's actual trajectory. Yet ADF is the inferred path, which is a 3-cell cut with the same CCL sequence as ADE. We refer to such a falsely inferred 3-cell cut, as a *False Line Segment (FLS)*. As FLSs degrade the inference accuracy, we propose a method to detect and remove the FLSs in $k : k'$-cell cut ($k, k' \geq 3$) trajectory segments.

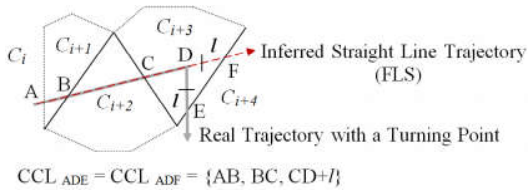


Figure 8: A false line segment inferred by TPDA with the same CCL sequence as a turning point trajectory

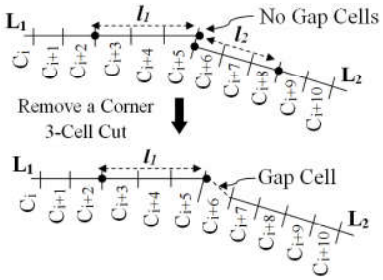


Figure 9: An unrealistic line segment combination resulted from an FLS, and the resulting line segment combination after the removal of FLS

Because of the sliding window 3-cell cut detection method in TPDA, FLSs inferred result in *Unrealistic Line Segment Combinations (ULSCs)*. As shown in the top diagram of Figure 9, a ULSC is the existence of two consecutive k-cell cuts ($k \geq 3$) such that the number of gap cells between them is less than one. Assume L_1, L_2 are two consecutive k-cell cuts ($k > 3$), inferred by TPDA, for cell sequence, C_{i+1}:C_{i+10}. l_1 and l_2 are the corner 3-cell cuts of L_1 and L_2. According to the diagram, no 3-cell cuts are detected in C_{i+4}:C_{i+6} and C_{i+5}:C_{i+7}, which shows that the trajectory within C_{i+6} is not a straight line. However, l_2 covers C_{i+6}:C_{i+8}, which shows the user's trajectory is a straight line within C_{i+6}. As these two are contradictory inferences, we consider this as a ULSC, caused by an FLS, and propose Algorithm 2 to remove the FLS.

Algorithm 2: Remove FLSs in different ULSCs

input : Line segments creating the ULSC (L_1, L_2)
output: Trajectory line segments (TLS)

```
1  NCC_L1, NCC_L2: NCC of L1, L2;
2  l1, l2: Corner 3-cell cut of L1, L2;
3  case NCC_L1, NCC_L2 > 3
4  |  L1 ← (L1 − l1) or L2 ← (L2 − l2);
5  |  TLS ← L1 and L2;
6  end
7  case NCC_L1 > 3, NCC_L2 = 3
8  |  Remove L2; TLS ← L1;
9  end
10 case NCC_L1, NCC_L2 = 3
11 |  Remove L1 or L2; TLS ← L2 or L1;
12 end
13 return TLS;
```

We show that when the real trajectory is a combination of k-cell cuts ($k > 3$), only the corner 3-cell cut of an inferred k-cell cut ($k > 3$) can be an FLS. Please refer to the link[4] for more details. Therefore, as shown in the bottom diagram of Figure 9, we remove either l_1 or l_2, reducing the NCC of L_1/L_2 (NCC_{L_1}/ NCC_{L_2}) by one, resulting in one gap cell between L_1 and L_2 (Step 3-6 in Algorithm 2). Furthermore, we show that in a $k : k'$-cell cut ($k, k' \geq 3$), a 3-cell cut has more probability of being an FLS (2/3 vs.1/3) than a k-cell cut ($k > 3$) containing an FLS. Please refer to the link [4] for more details. Therefore, in the above combination, if $NCC_{L_1} > 3$ and $NCC_{L_2} = 3$, we always consider L_2, as the FLS and completely remove it (Step 7-9 in Algorithm 2). If $NCC_{L_1} = NCC_{L_2} = 3$, we randomly remove a one (Step 10-12 in Algorithm 2).

4. EXPERIMENTAL STUDIES

We evaluate the accuracy of our algorithm under various experimental conditions for both synthetic and real data.

4.1 Data Sets

4.1.1 Synthetic Data Generation

We select a geographical area of 50 km × 50 km in Melbourne area. The cell tower locations are assumed to be

[4]http://people.eng.unimelb.edu.au/baileyj/sigspatial_supp.pdf

uniformly distributed within this area. A Voronoi diagram is created considering these points as Voronoi cell centres, simulating the cellular network. To vary the cell density in the range of 0.4 – 4 towers/km^2, we vary the number of cell towers from 1000 – 10000. The maximum and minimum cell densities represent typical urban and suburban cell densities as obtained from Australian Geographical RadioFrequency Map[5]. The start point of a trajectory is randomly selected, assuming uniform distribution within the selected geographical area. The trajectory is then generated as a line segment combination, demonstrating a given parameter value combination (see Section 4.3.1). ~40K random trajectories are generated, 1000 for a given parameter value in Table 2. The latitude-longitude sequences of the trajectories are considered as the ground truth trajectories. The cell centres of the Voronoi cell sequence, intersected by the trajectory, are considered as the cellular trajectory of the user. The length of ground truth trajectory segments lying within the boundaries of a Voronoi cell, is computed as the CCL of that cell.

4.1.2 Real Data Generation

People's everyday trajectory patterns are selected from two geographical areas: 1) an urban area around Melbourne Central Business District and 2) a suburban area in Melbourne (see Section 4.4.1). A user, with a smart phone, made several trips, tracing a given trajectory pattern. For the convenience of measurements, a known constant speed was maintained. The phone's connected cell tower IDs and corresponding time stamps are collected by a software application, installed in the user's smartphone. Fourteen such cellular trajectories are obtained for seven different patterns, as shown in Table 3. The latitude-longitude sequence of the road segments, traversed by the user, are obtained from Google Maps [6]. This is considered as the ground truth trajectory. The cell handover times are computed using time stamp information. The CCLs are computed using the average user speed and handover time information.

4.2 Evaluation Metric and Criteria

In this section, we describe the evaluation metric, the baseline method and the trajectory features against which our algorithm is evaluated.

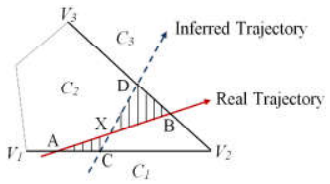


Figure 10: Average error computation

4.2.1 Evaluation Metric

We evaluate the accuracy of our algorithm in terms of *Average Error (AE)*. This measures the spatial similarity between the inferred and the real trajectory within a cell and is computed as the average deviation between the two. The Error Area per Cell (EAC) is the area formed by the two poly lines (real trajectory and inferred trajectory) on the 2-D plane and the cell boundaries as shown by the shaded area in Figure 10. Thus, the AE is computed as:

$$AE = \sum EAC / Real\ Trajectory\ Length \quad (2)$$

4.2.2 Baseline Method

The existing cellular trajectory inference techniques either use a significant amount of knowledge about the cellular network infrastructure or have limitations to use in a device centric manner due to historical trajectory information and pre-training requirements [1, 15, 22]. Hence, we cannot directly compare our method against those techniques. Therefore, we suggest a baseline trajectory inference method which only uses the connected cell tower information as our approach does. As shown in Figure 11, a baseline trajectory is obtained by connecting the cell centers of the consecutive cells of the cellular trajectory with straight line segments.

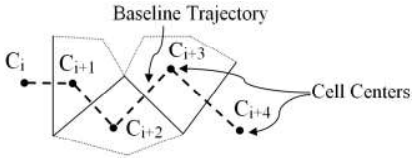


Figure 11: Baseline method for trajectory inference

4.2.3 Trajectory Features Evaluated

In this section, we discuss the trajectory features which affect our algorithm's accuracy and define the parameters to measure them. We use both meters/kilometers and NCC to measure (or estimate) the lengths of the trajectory line segments. We use *Turn Frequency* for trajectories with a uniform turn frequency, i.e., having same distance between each turning point. The *Maximum Turn Frequency*, *Short Segment Percentage* and *Short Segment Distribution* are used for trajectories with non-uniform turn frequencies.

1. **Total Trajectory Length (TTL):** This is computed as the total distance, a user has travelled and is measured in *meters* (considering geographical length) or in *Total Cells Traversed (TCT)*, i.e., the total number of cells completely covered by the trajectory (Considering NCC).
2. **Turn Frequency (TF):** It is the number of turns taken for a unit distance and computed as *1/ Distance between Two Turning Points*. The measurement unit of TF is *number of turns per meter* or *number of turns per cell*.
3. **Maximum Turn Frequency (MTF):** This is the maximum TF value in the trajectory and is computed as *1/ Shortest Distance between Two Turning Points*.
4. **Short Segment Percentage (SSP):** This is the percentage of the trajectory length which exists as *Short Segments*. A *Short Segment* is a k-cell cut ($k < 3$) which cannot be exactly inferred by the algorithm. SSP is computed as *Total Trajectory Length within Short Segments * 100)/ TTL*.
5. **Short Segment Distribution (SSD):** If short segments and longer segments are separated from each other in the trajectory, we say that the trajectory's SSD is *"Clustered"*. Otherwise it is *"Distributed"*.
6. **Cell Density (CD):** This is the number of cell towers of the cellular network distributed within a unit area.
7. **Average Speed (AS):** This is the average user speed and is computed as *TTL/Total Time for the Journey*.

[5]http://maps.spench.net/rf
[6]https://maps.google.com

Table 2: Parameter values used in synthetic trajectory generation (~40K trajectories)

Experiment #	CD (Towers per km^2)	TTL (TCT)	TF(Turns per Cell)	SSP (%)	MTF(Turns per Cell)	SSD(Clustered/ Distributed)
1	0.4	**1 – 20**	0	N/A	N/A	N/A
2	0.4	20	**1/5 – 1**	N/A	N/A	N/A
3	0.4	20	N/A	**0 – 100**	**1/2**	Distributed
					1	Distributed
4	0.4	20	N/A	**0 – 100**	1/2	**Clustered**
						Distributed
5	**0.4 – 4**	~15 (km)	1 (Turns per km)	N/A	N/A	N/A

4.3 Experiments for Synthetic Data

The algorithm's accuracy is evaluated against the trajectory features discussed above, using the synthetic data set.

4.3.1 *Experimental Setup*

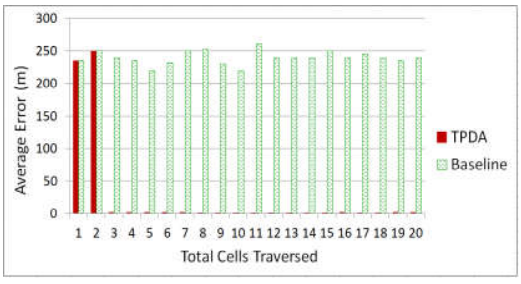
Trajectories are created assigning different values for each of the above parameters (in Section 4.2.3) as mentioned in Table 2. For a given parameter value, 1000 random trajectories are created and we report the average AE value. Baseline method is also implemented on the same dataset. The main purpose of experiments 1 – 4 is to analyse the effect of NCC of line segments on the algorithm's accuracy. The line segment lengths are thus estimated in NCC. In Experiment 5, the segment lengths are measured in kilometers.

4.3.2 *Experimental Results for Synthetic Data*

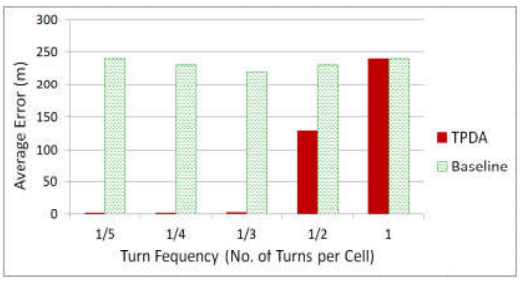
Figure 12 describes the results obtained for experiments using the synthetic dataset. Each graph represents the error reported by both the proposed (TPDA) and the baseline methods for each experiment summarised in Table 2.

As shown in Figure 12a, in Experiment 1, the error significantly decreases (~1 m) for 3-cell cuts compared to the baseline error (~240 m) and remains constant for longer cell cuts. Figure 12b (Experiment 2) shows that the $AE_{TF=1} > AE_{TF=1/2} > AE_{TF\leq 1/3}$, i.e., the error decreases with the decrease of TF and remains constant after 1/3 turns per cell. For any TF $\leq$ 1/3, the error is significantly low (~1 m) compared to the baseline error (~240 m) and error for TF=1. Figure 12c (Experiment 3) shows that the error increases with the increase of SSP. At high SSPs, trajectories with 2-cell cut short segments report a lower error (~120 m), compared to the error reported by trajectories with 1-cell cut short segments (~240 m). Results in Experiment 2 show that trajectories whose MTF $\leq$ 1/3 reports significantly lower error (~1 m), compared to trajectories whose MTF > 1/3 (~240 m, ~120 m). In Experiment 3, the trajectories whose minimum line segment length is two cell cuts (MTF=1/2) produce lower error than the trajectories whose minimum line segment length is one cell cut (MTF=1). These observations show that the error increases with the increase of MTF. Figure 12d (Experiment 4) shows that for each SSP, the clustered trajectories report a higher error than the distributed trajectory set, yet this is only a slight difference. At low SSP levels (0 – 15%), both trajectories show a lower error(~1 – 10 m) compared to the baseline error (~240 m).

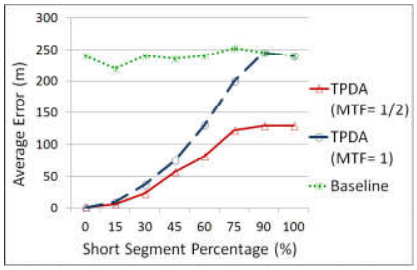
Figure 12e (Experiment 5) shows that for a given spatial trajectory (latitude-longitude sequence), the error decreases with the increase in CD. The maximum and minimum CDs



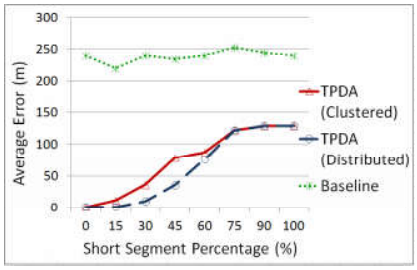
(a) Average error variation with trajectory length



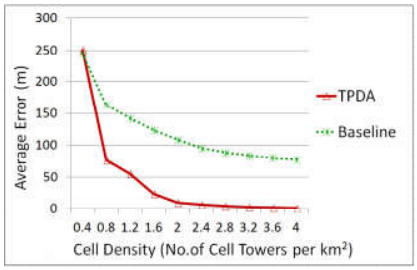
(b) Average error variation with turn frequency



(c) Average error variation with short segment percentage



(d) Average error variation for different short segment distributions



(e) Average error variation with cell density

Figure 12: Average error variations for different trajectory and cellular network parameter values

Table 3: Everyday trajectory patterns selected for the real data set and their parameter values

Pattern	Description	Travel Mode	Area	ATF	TTL
1	Tram trip between university and railway station	Tram	Urban	Low (0.5 km^{-1})	~2 km
2	Walk between university and railway station	Walk	Urban	Low (0.5 km^{-1})	~2 km
3	Tram trip between apartment and railway station	Tram	Urban	Low (0.5 km^{-1})	~2 km
4	Walk between university and apartment	Walk	Urban	Low (6.67 km^{-1})	~0.6 km
5	Shopping trip	Walk	Urban	High (3.25 km^{-1})	~4 km
6	Train trip in an urban area	Train	Urban	High (Curved Roads)	~4 km
7	Train trip in a suburb	Train	Suburban	High (Curved Roads)	~20 km

represent typical urban and suburban cell densities. Baseline error decreases (~250 – ~75 m) with increasing cell density (0.4 – 4 towers/km^2), however, at a much lower rate compared to TPDA (~250 – ~1 m).

4.4 Experiments for Real Data

We evaluate the accuracy of TPDA on real world trajectories to analyse its applicability in real world scenarios.

4.4.1 Experimental Setup

We perform experiments using fourteen trajectories, reflecting the patterns described in Table 3. These trajectory patterns are selected to represent the everyday behaviour of people and also include the different trajectory features as discussed in Section 4.2.3. Patterns 1-6 are produced in the urban area and Pattern 7 in the suburban area, as described in Section 4.1.2. This is to ensure that they represent different CDs, ~20 cell towers/ km^2 and ~0.4 cell towers/ km^2 for urban and suburban areas, respectively. The long and short trajectories, (Pattern 2 and Pattern 4) are produced to represent different TTLs. The trajectories with high and low turn frequencies are produced in Pattern 2 and Pattern 5, respectively, representing different TFs. These trajectories contain multiple turn frequencies. As MTF, SSP, and SSD are for NCC based evaluation, and are of less importance from a user's perspective, we eliminate them in real data experiments. Instead, we introduce the measure, *Average Turn Frequency (ATF)*, the *average number of turns per unit distance*, computed as *# Total Turning Points/TTL*. To evaluate the effect of user speed, we consider travel patterns using the transportation modes – walking, tram and train with the AS values of ~1 ms^{-1}, ~3 ms^{-1} and ~10 ms^{-1}.

4.4.2 Experimental Results for Real Data

Figure 13 describes the experimental results obtained for real trajectory patterns. Pattern 1 and 3 are long trajectories (~2 km) with low turn frequencies. For them, TPDA reports significantly lower errors (~7 m) compared to the baseline errors (~30 m and ~63 m). Figure 14 shows the graphical output of ground truth and inferred trajectories for Pattern 1, plotted on the cellular network of the area. Pattern 2, following the same route as Pattern 1 at a lower speed, has lower error (~1 m) than Pattern 1 (~7 m). For Pattern 4, which is a short trajectory, TPDA reports a high error same as the baseline error (~130 m). Pattern 5, which has high turn frequency, reports a higher error (~17 m) than low turn frequency trajectory patterns such as Pattern 1 (~7 m) and Pattern 2 (~1 m). In Pattern 6, the user travels at a high speed and the error reported is as high as the baseline error (~80 m). Pattern 7 is a trajectory from a suburban area with low cell tower density and curved trajectory segments. It reports a high error (~145 m) same as the baseline method. Both errors are significantly high compared to other trajectory patterns from the urban area.

Visible Cell Density Effect: The results show that higher user speeds result in a higher error, even though the same route is followed (Pattern 1 vs. Pattern 2). This is due to travelling a large distance within a very small time, the user does not have chance to connect to all the cell towers whose Voronoi cells are intersected by the trajectory. Therefore, the cell density visible to the user is different from the real cell density. For example, in Pattern 1 when travelling by tram, the visible cell density is ~4 cell towers/km^2 (~500 m distance between two cell towers) even though the real cell density is ~20 cell towers/km^2 (~150 m distance between two cell towers). However, in Figure 13, Pattern 1 has ~7 m average error. This demonstrates that unseen cell towers do not fundamentally affect TPDA's accuracy.

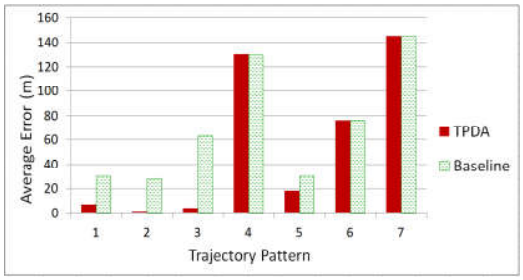


Figure 13: Average errors obtained for different trajectory patterns in the real data set

Imprecise Speed Measurements: We evaluate the algorithm accuracy for imprecise speed measurements, as follows. Traversed Pattern 2/Pattern 3 with a varying speed in the range 0 – 1.5 ms^{-1}/0 – 4 ms^{-1}, and computed the average speed for the journey. CCLs are computed assuming the user has maintained this average speed throughout the journey. The trajectory is inferred by setting δd to 20 m. Pattern 2 reports an error, ~32 m slightly higher than the baseline error, ~28 m. Pattern 3 reports a lower error, ~48 m than the baseline error, ~60 m. The results show that TPDA is robust enough to handle imprecise speed information.

4.4.3 Key Evaluation Insights

Our experiments with synthetic and real data demonstrate that TPDA achieves high accuracy, which is same as GPS, for straight line trajectories with low turn frequencies. As shown in Figure 14, the exact trajectory segments are inferred when they are at least 3-cell cuts, producing a nearly zero error. TPDA achieves very high accuracy with the in-

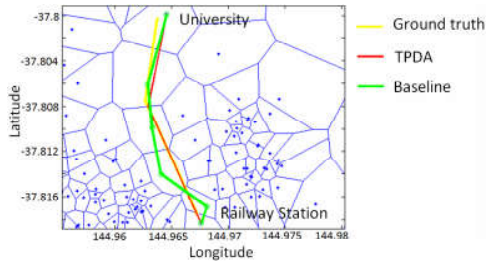


Figure 14: Ground truth and inferred trajectories for Pattern 1, plotted on the cellular network

crease of cell density. Therefore, it can be successfully applied in urban areas with straight line road segments and high cell densities, benefitting more in case of 4G networks. However, for high turn frequencies, low cell densities and curved trajectories, it shows a lower accuracy than GPS. The accuracy is not affected by trajectory length beyond three cell cuts. Furthermore, we show that unseen cell towers do not substantially affect the accuracy and TPDA is robust enough to handle imprecise speed information.

5. CONCLUSIONS

In this paper, we propose a device centric cellular trajectory inference method using only the user's connected cell tower locations, handover times and speed information. The proposed method does not require any historical trajectory information or pre-training and incurs low storage and computation costs. It exactly infers the line segments of a user's trajectory and approximates other segments accordingly. Our experimental results with both synthetic and real data confirm that our algorithm produces significantly higher accuracy compared to the baseline method when the user does not change direction frequently in urban areas with a high cell density. For the occasions the trajectory is a line segment combination with turning points, each line segment covering a minimum of three cells, the trajectory is exactly inferred, achieving a nearly zero error. However, in suburbs with low cell densities and curved roads, it reports an error, same as the baseline error, ~145 m. We also show that our method is robust enough to handle imprecise speed information due to user speed variations and unseen cell towers do not affect its accuracy. In future, we will focus on integrating road network information to improve the algorithm's accuracy and using the algorithm interchangeably with GPS such that GPS has to be only used when required.

6. REFERENCES

[1] M. S. Arulampalam et al. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 2(2):174–188, 2002.

[2] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM CSUR*, 23(3):345–405, 1991.

[3] P. Brida et al. An Experimental Evaluation of AGA Algorithm for RSS Positioning in GSM Networks. *Electronics and Electrical Engineering*, 104(8):113–118, 2010.

[4] M. Y. Chen et al. Practical Metropolitan-Scale Positioning for GSM Phones. *Ubiquitous Computing*, 4206:225–242, 2006.

[5] C. Drane et al. Positioning GSM Telephones. *Communications Magazine, IEEE*, 36(4):46–54, 59, 1998.

[6] M. Ficek et al. Performance Study of Active Tracking in a Cellular Network Using a Modular Signaling Platform. In *Proc. of 8^{th} International Conference on Mobile Systems, Applications, and Services*, pages 239–254, Kraków, Poland, 2010.

[7] S. Fortune. A Sweepline Algorithm for Diagrams. *Algorithmica*, 2:153–174, 1987.

[8] S. Fortune. Voronoi Diagrams and Delaunay Triangulations. *Computing in Euclidean geometry*, 1:193–233, 1992.

[9] S. Helhel et al. Investigation of GSM Signal Variation Dry and Wet Earth Effects. *Progress in Electromagnetics Research*, 1:147–157, 2008.

[10] A. Hussain et al. Positioning in Wireless Body Area Network using GSM. *JDCTA*, 2009.

[11] Khan. J. Handover Management in GSM Cellular System. *International Journal of Computer Applications*, 8(12), 2010.

[12] A. LaMarca et al. Place lab: Device Positioning using Radio Beacons in the Wild. In *Pervasive computing*, pages 116–133. Springer, 2005.

[13] L. Mihaylova et al. Mobility Tracking in Cellular Networks Using Particle Filtering. *IEEE Transactions on Wireless Communications*, 6(10):3589–3599, 2007.

[14] M. Sharifzadeh et al. Approximate Voronoi Cell Computation on Spatial Data Streams. *The VLDB Journal*, 18:57–75, 2009.

[15] A. Thiagarajan et al. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *NSDI*, Boston, 2011.

[16] E. Trevisani et al. Cell-ID Location Technique, Limits and Benefits: an Experimental Study. In *Proc. of 6^{th} IEEE WMCSA*, pages 51–60, 2004.

[17] A. Varshavsky et al. Are GSM Phones THE Solution for Localization? In *Proc. of 7^{th} IEEE WMCSA*, pages 34–42, 2006.

[18] A. Varshavskya et al. GSM Indoor Localization. *Pervasive and Mobile Computing*, 3(6):698–720, 2007.

[19] A. Waadt et al. Traffic Congestion Estimation Service Exploiting Mobile Assisted Positioning Schemes in GSM Networks. *Procedia Earth and Planetary Science*, 1(1):1385–1392, 2009.

[20] J. Yang et al. Accuracy Characterization of Cell Tower Localization. In *Proc. of Accurate, Low-Energy Trajectory Mapping for Mobile Devices12th ACM International Conference on Ubiquitous Computing*, pages 223–226, Copenhagen, Denmark, 2010.

[21] Y. Yuan et al. Extracting Dynamic Urban Mobility Patterns from Mobile Phone Data. In *GIScience*, volume 7478, pages 354–367. Springer, 2012.

[22] Z. R. Zaidi et al. Real-Time Mobility Tracking Algorithms for Cellular Networks Based on Kalman Filtering. *IEEE Transactions on Mobile Computing*, 4(2):195–208, 2005.

[23] J. Zhang et al. Location-based Spatial Queries. In *Proc. of 2003 ACM SIGMOD*, pages 443–454, San Diego, 2003.