CONTENTS

Efficient Contact Similarity Query over Uncertain Trajectories

Xichen Zhang, Suprio Ray, Farzaneh Shoeleh, Rongxing Lu
Canadian Institute for Cybersecurity, Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada
{xichen.zhang,sray,farzaneh.shoeleh,rlu1}@unb.ca

ABSTRACT

The problem of studying the effective contact between different moving objects (MOs) has received great interest in recent years. However, the uncertain nature of trajectory data poses significant challenges for the researchers. Most of the existing studies focus on range query or similarity join. But they cannot find MOs who may contact each other for a long enough time in the same location. In this paper, we study how to evaluate the effective contact among MOs. More specifically, we provide a purposeful definition for measuring the contact effectiveness, called **Contact Similarity**. Based on this notion, we introduce a novel query called **Contact Similarity Query** (*CSQ*). A necklace-based model is used for representing uncertain trajectories, in which a bead (or an ellipse) indicates the possible locations of an object in-between two successive trajectory observations. To enable efficient query processing, we design a novel data structure called UTM-tree (i.e., Uncertain Trajectory M-tree) for indexing the necklace-based trajectories. Experiments have been conducted on a real-world dataset, and the results demonstrate that our proposed solution significantly outperforms the baseline approaches.

1 INTRODUCTION

With the increasing availability and rapid development of global positioning technologies, moving objects (MOs)' trajectories can be utilized by Location-Based Services (LBS) in many applications, such as vehicle navigation, traffic management, and co-occurrence analysis. However, due to sensor devices' physical and resource limitations or privacy considerations, MOs' trajectories are often captured at low sampling rates, and the time interval between two consecutive observations is quite long. In such uncertain courses, no information can be found about the whereabouts of MOs in-between two successive points. Recently, the problem of querying uncertain trajectory has been studied by many works, e.g., [1, 2, 5, 6, 10, 13]. However, most of them focused on retrieving qualified results regarding either probabilistic range query or similarity join. The existing approaches are not applicable to the problem of modeling the effective contact among individuals, which is a research topic of great importance. As follows, a real-world example is presented to describe a motivational scenario.

Example 1. *Fig. 1 shows the uncertain trajectories of three visitors m_1, m_2, and m_3 in one day. They are equipped with a GPS device in their car, and they periodically reported their real-time locations along their trajectories. But we don't know where they are in-between two consecutive reports. We can see that m_1 and m_3 may have a very high trajectory similarity since the spatial and temporal distance between their observations can be very small. However, m_1 and m_3 may never come in contact with each other. On the other hand, assume that along the trip, m_1 parked the car at location A, disembarked the car, walked into the Park and stayed there for a while. Then m_1 went back to A to board the car and continued with the trip. Perhaps m_2 can take similar action at location B, i.e., m_2 may also stay in the Park for a while during his/her trip. Therefore, even though m_1 and m_2 have very low trajectory similarity, they may have an effective contact either in the Park or in the Shopping District. But such a result can never be captured by the traditional trajectory similarity query. Besides, assume that several days later, m_1 found him/herself as an infected individual of a highly contagious disease (e.g., Covid 19). To help determine other susceptible persons who might be infected, m_1 provided more details about his/her trajectory to an authorized third-party, such as a local public health agency (LPHA). For example, m_1 stayed in the Park from 10:00 am to 10:50 am, and then went to the Shopping District from 11:35 am to 12:30 pm. Next, given this information, LPHA discovered that m_1 and m_2 may have possible contact with each other, and may like to know the effectiveness of their contact (e.g., the chance of their contact and how long they may contact with each other). Moreover, hundreds of people may have visited the neighboring areas on the same day. LPHA wants to find the susceptible individuals from the big group of candidates who may have effective contact with m_1.*

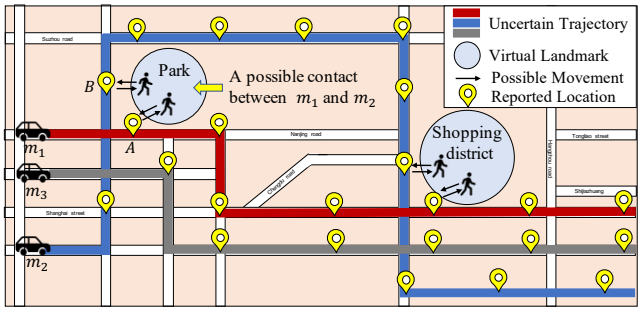


Figure 1: An real-world example of possible contacts between MOs m_1, m_2, and m_3

To address the problem described above, it is important to answer the following questions successively:

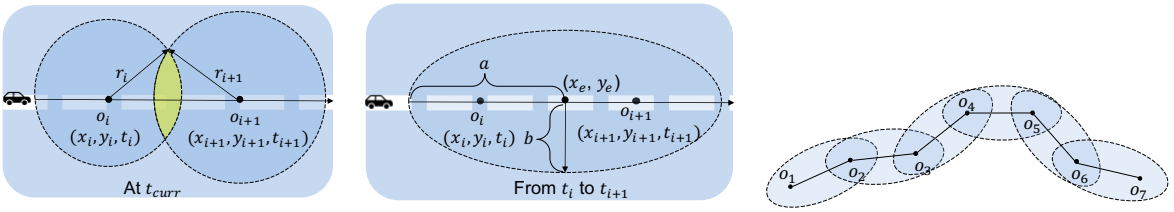
Q1: How to calculate the potential contact area and the possible longest contact time duration between m_1 and m_2?

Q2: How to measure the effective contact between m_1 and m_2?

Q3: How to efficiently retrieve the qualified results from a large number of people who visited the park on the same day as m_1?

To the best of our knowledge, little or no systematic and theoretical study has been conducted to address the abovementioned questions. Aiming to fill this research gap, we propose a study on analyzing the effective contact between MOs over uncertain trajectories. Specifically, the main contributions of this work are three-fold as follows.

• We propose a definition of **Contact Similarity** over uncertain trajectories and a novel query called **Contact Similarity Query** (*CSQ*). With *CSQ*, we can assess the effectiveness of contact between MOs in free space.



(a) The possible locations of the MO at time $t_{curr}(t_i \le t_{curr} \le t_{i+1})$

(b) All the possible locations (the bead B_i) of the MO between t_i and t_{i+1}.

(c) An example of necklace for a given uncertain trajectory with 7 observations.

Figure 2: An example of bead B_i given the information of o_i, o_{i+1}, and v_{max}.

Table 1: The summary of notations

Notation	Definition
Trajectory Notations	
MO	Moving object
$\mathsf{T} = \{o_1, o_2, ..., o_n\}$	An uncertain trajectory
$t_i/(x_i, y_i)$	Timestamp/coordinates of o_i in T
v_{max}	The maximum speed of a MO
T_{neck}	The necklace of T
B_i	A bead in T_{neck}
P	A virtual landmark
t^s/t^e	The arrival/departure time of P
Query Notations	
R_{int}	The intersection area ratio
$\tau(\mathsf{P}, \mathsf{B}_i)$	The possible longest time duration
$Sim_{cont}^{\mathsf{P} \cap \mathsf{B}_i}$	Contact similarity at bead level
D	A MO database
$Sim_{cont}^{\mathsf{P} \cap \mathsf{T}}$	Contact similarity at trajectory level
$\bar{t}^s$	The earliest arrival time
$\bar{t}^e$	The latest departure time

• We design a novel data structure called UTM-tree (Uncertain Trajectory M-tree) for indexing uncertain trajectories, which significantly improves the efficiency for processing *CSQ*.

• We conduct an extensive experimental evaluation of the proposed approaches, demonstrating the effectiveness and efficiency of the methodology proposed in this work.

The rest of this paper is organized as follows. In Section 2, we introduce the preliminaries and problem definition. In Section 3, we propose the novel indexing structure for managing uncertain trajectories. In Section 4, we present the details of our experiments and the evaluation results, followed by the related works in Section 5. Finally, we recap the conclusions in Section 6.

2 PRELIMINARIES

In this section, we briefly introduce the preliminaries of this work. The frequently used notations are listed in Table 1.

2.1 Trajectories, Beads, and Necklaces

We first introduce the definitions of trajectory, beads, and necklaces. A trajectory is defined as follows.

Definition 1. *A **trajectory** T of a MO is defined as finite, time-ordered observations $\mathsf{T} = \{o_1, o_2, ..., o_n\}$, where $o_i = (x_i, y_i, t_i)$ for $i \in [1, n]$, with (x_i, y_i) being a sample point in Euclidean space, and t_i being a timestamp.*

The possible locations of a MO in-between two observations can be defined by a bead (or ellipse) [10] as follows.

Definition 2. *Let v_{max} denote the maximum speed that an object can take between o_i and o_{i+1}. A **bead** $\mathsf{B}_i = \{(x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1})\}$ is defined as all points (x, y, t) that satisfy the following constraints:*

$$\begin{cases} t_i \le t \le t_{i+1} \\ (x - x_i)^2 + (y - y_i)^2 \le (t - t_i)^2 \times v_{max}^2 \\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \le (t_{i+1} - t)^2 \times v_{max}^2. \end{cases} \tag{1}$$

From Fig. 2 (a), we can see that from time t_i to t_{curr} ($t_{curr} > t_i$), the MO's possible travel area is a circle with (x_i, y_i) being the center and $r_i = (t_{curr} - t_i) \times v_{max}$ being the radius. Similarly, from time t_{curr} to t_{i+1} ($t_{curr} < t_{i+1}$), the possible locations are included in a circle centered at (x_{i+1}, y_{i+1}) with radius $r_{i+1} = (t_{i+1} - t_{curr}) \times v_{max}$. So, the overlapped area in Fig. 2 (a) of the two circles includes MO's possible locations at current time t_{curr}. Based on [8], all the possible locations from time t_i to t_{i+1} form an ellipse (or bead) with foci at (x_i, y_i) and (x_{i+1}, y_{i+1}) (see in Fig. 2 (b)). According to [10], the equation of the bead B is:

$$(x - x_e)^2/a^2 + (y - y_e)^2/b^2 = 1, \tag{2}$$

where (x_e, y_e) is the center of the ellipse; a and b represents the semi-major axis and semi-minor axis of the ellipse, as follows.

$$x_e = \frac{x_i + x_{i+1}}{2},\ y_e = \frac{y_i + y_{i+1}}{2},\ a = \frac{v_{max} \times (t_{i+1} - t_i)}{2}$$
$$b = \frac{\sqrt{v_{max}^2 \times (t_{i+1} - t_i)^2 - (x_{i+1} - x_i)^2 - (y_{i+1} - y_i)^2}}{2}. \tag{3}$$

Definition 3. *A trajectory $\mathsf{T} = \{o_1, o_2, ..., o_n\}$ can be represented as a sequence of beads, which is called a **necklace** and denoted as $\mathsf{T}_n = \{\mathsf{B}_1, \mathsf{B}_2, ..., \mathsf{B}_{n-1}\}$ (see an example in Fig. 2 (c)), such that o_i and o_{i+1} form the bead B_i for $i \in [1, n-1]$.*

2.2 Contact Similarity

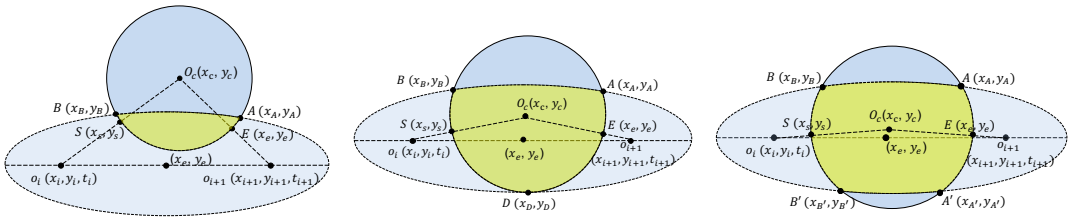
In this section, we introduce the essential definitions for contact similarity.

Definition 4. *A **virtual landmark** P is a place where two MOs can contact with each other. P can be either meaningful locations (e.g., a shopping district or a park) or non-meaningful locations (e.g., a crossroad). For simplicity sake, any P in this work is considered as a circle that is centered at $O_c = (x_c, y_c)$ with a given radius r_c. Hence, the equation of P is $(x - x_c)^2 + (y - y_c)^2 = r_c^2$.*

Definition 5. *For a MO m_1, he/she stayed at P for a certain period. For another MO m_2, his/her necklace is $\mathsf{T}_n = \{\mathsf{B}_1, \mathsf{B}_2, ..., \mathsf{B}_{n-1}\}$. If P spatially overlaps with one bead $\mathsf{B}_i \in \mathsf{T}_n$ for $i \in [1, n-1]$ (denoted as $\mathsf{P} \cap_S \mathsf{B}_i \neq \emptyset$), then the **intersection area ratio** R_{int} is*

$$\mathsf{R}_{int} = \mathsf{Area}_{int}/\mathsf{Area}_P, \tag{4}$$

where Area_{int} is the intersection area between the P and B_i, and Area_P is the area of P, i.e., $\mathsf{Area}_P = \pi \cdot r_c^2$.



(a) *Two* intersection points (A, B) (b) *Three* intersection points (A, B, D) (c) *Four* intersection points (A, B, A', B')

Figure 3: Examples of overlap between a virtual landmark and a bead with different intersection points

Fig. 3 shows different scenarios that P (the circle) overlaps with a bead B_i (the ellipse), with two, three, and four intersection points, respectively. Then, we difine the possible longest contact time duration between MOs as follows.

Definition 6. *For MO m_1, he/she stayed at P from time t^s to t^e. For another MO m_2, his/her necklace $\mathsf{T}_n = \{\mathsf{B}_1, \mathsf{B}_2, ..., \mathsf{B}_{n-1}\}$. Moreover, if one of the beads $\mathsf{B}_i \in \mathsf{T}_n$ (i.e., $i \in [1, n-1]$) temporally overlaps with P (denoted as $\mathsf{P} \cap_T \mathsf{B}_i \neq \emptyset$), then the **possible longest time duration** $\tau(\mathsf{P}, \mathsf{B}_i)$ between m_1 and m_2 can be calculated as*

$$\tau(\mathsf{P}, \mathsf{B}_i) = \min\{t^e, \bar{t}^e\} - \max\{t^s, \bar{t}^s\}, \tag{5}$$

where $\bar{t}^s$ means the possible earliest time for m_2 to arrive at P, and $\bar{t}^e$ means the possible latest time for m_2 to leave P.

Example 2. *If m_1 stayed at P from t^s = 10:00 am and t^e = 10:50 am. For m_2, $\bar{t}^s$ = 10:05 am and $\bar{t}^e$ = 10:35 am, then $\tau(\mathsf{P}, \mathsf{B}_i)$ = min{10:50, 10:35} - max{10:00, 10:05} = 30 min.*

Next, inspired by [7], we formally propose the definition of contact similarity as below.

Definition 7. *For m_1, he/she stayed at P from time t^s to t^e. For m_2, his/her necklace is $\mathsf{T}_n = \{\mathsf{B}_1, \mathsf{B}_2, ..., \mathsf{B}_{n-1}\}$. If $\exists \mathsf{B}_i \in \mathsf{T}_n$ such that $\mathsf{P} \cap_S \mathsf{B}_i \neq \emptyset$ and $\mathsf{P} \cap_T \mathsf{B}_i \neq \emptyset$, then **the contact similarity at bead level** $Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i}$ between m_1 and m_2 at P in terms of B_i can be calculated as:*

$$Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i} = 1 - (1 - \mathsf{R}_{int})^{\tau(\mathsf{P}, \mathsf{B}_i)} \tag{6}$$

We assume that the intersection area ratio R_{int} indicates the possible opportunities that m_1 may be in contact with m_2 at P per unit time. Therefore, $(1 - \mathsf{R}_{int})^{\tau(\mathsf{P}, \mathsf{B}_i)}$ denotes the probability that m_1 and m_2 have no effective contact during the time period $\tau(\mathsf{P}, \mathsf{B}_i)$. Consequently, $Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i}(m_1, m_2) = 1 - (1 - \mathsf{R}_{int})^{\tau(\mathsf{P}, \mathsf{B}_i)}$. A larger value of $Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i}(m_1, m_2)$ indicates a higher probability that an effecitive contact may happen between m_1 and m_2 at P.

Definition 8. *Let $\mathbb{B}$ denote a set of beads in T_n, such that for $\forall \mathsf{B}_i \in \mathbb{B}$, $\mathsf{P} \cap_S \mathsf{B}_i \neq \emptyset$ and $\mathsf{P} \cap_T \mathsf{B}_i \neq \emptyset$. Then, **the contact similarity at trajectory level** $Sim_{cont}^{\mathsf{P}\cap\mathsf{T}}$ can be calculated as:*

$$Sim_{cont}^{\mathsf{P}\cap\mathsf{T}} = \frac{\sum_{\mathsf{B}_i \in \mathbb{B}} Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i}}{|\mathbb{B}|}. \tag{7}$$

Here, $|\mathbb{B}|$ denotes the number of beads in $\mathbb{B}$.

2.3 Problem Definition

Definition 9. *For m_1, the **query trajectory** T_q is defined as a sequence of P(s) that m_1 visited, i.e., $\mathsf{T}_q = \{\mathsf{P}_1, \mathsf{P}_2, ..., \mathsf{P}_d\}$. $\mathsf{P}_k = (x_k^c, y_k^c, r_k^c, t_k^s, t_k^e)$ for $k \in [1, d]$. Here, m_1 stayed at P_k from t_k^s to t_k^e. (x_k^c, y_k^c) and r_k^c are the center and radius of P_k, respectively.*

Definition 10. *Given a query trajectory $\mathsf{T}_q = \{\mathsf{P}_1, \mathsf{P}_2, ..., \mathsf{P}_d\}$, a MO database D that contains s necklaces (e.g., $D = \{\mathsf{T}_{n1}, \mathsf{T}_{n2}, ..., \mathsf{T}_{ns}\}$), and a predefined threshold α, for each $\mathsf{P}_k \in \mathsf{T}_q$ where $k \in [1, d]$, the **Contact Similarity Query** (CSQ) finds all the necklaces $\mathsf{T}_{nj} \in D$ for $j \in [1, s]$, such that $Sim_{cont}^{\mathsf{P}_k\cap\mathsf{T}_{nj}} \geq \alpha$.*

Example 3. *In Fig. 1, T_q is m_1's query trajectory. For example, m_1 stayed in the park P_1 from 10:00 to 10:50 am, then m_1 went to the shopping district P_2 and stayed there from 11:35 am to 12:30 pm. D is a MO database that contains 1000 people's trajectory necklaces who visited the neighboring areas on the same day, and let $\alpha = 0.8$. For every virtual landmark P_k that m_1 visited, the CSQ finds all the trajectory necklaces $\mathsf{T}_{nj} \in D$ such that $Sim_{cont}^{\mathsf{P}_k\cap\mathsf{T}_{nj}} \geq 0.8$.*

2.4 Calculation of R_{int} and $\tau(\mathsf{P}, \mathsf{B}_i)$

In this section, we discuss how to calculate R_{int} and $\tau(\mathsf{P}, \mathsf{B}_i)$, followed by the workflow for *CSQ*.

Given the P with equation $(x - x_c)^2 + (y - y_c)^2 = r_c^2$ and the B_i with equation $(x - x_e)^2/a^2 + (y - y_e)^2/b^2 = 1$, it is straightforward to calculate R_{int} as follows.

• There are two or three intersection points between the virtual landmark P and the bead B_i (see Fig. 3 (a) and (b)).

$$\mathsf{R}_{int} = \int_{x_B}^{x_A} \left(\frac{b}{a}\sqrt{a^2 - (x - x_e)^2} + y_e + \sqrt{r_c^2 - (x - x_c)^2} - y_c\right)dx \tag{8}$$

• There are four intersection points between the virtual landmark P and the bead B_i (see Fig. 3 (c)).

$$\begin{aligned}\mathsf{R}_{int} =& \pi r_c^2 - \int_{x_B}^{x_A} \left(\sqrt{r_c^2 - (x - x_c^2)} + y_c - \frac{b}{a}\sqrt{a^2 - (x - x_e)^2} - y_e\right)dx \\ & - \int_{x_B'}^{x_A'} \left(y_e - \frac{b}{a}\sqrt{a^2 - (x - x_e)^2} + \sqrt{r_c^2 - (x - x_c)^2} - y_c\right)dx\end{aligned} \tag{9}$$

In Fig. 3, we can easily know that S is the earliest point that one MO can arrive at P, and E is the latest point that MO can leave P. Therefore, given the maximum speed v_{max}, we can calculate $\bar{t}^s$ and $\bar{t}^e$ as follows.

$$\bar{t}^s = t_i + \frac{|o_i S|}{v_{max}},\ \bar{t}^e = t_{i+1} - \frac{|o_{i+1} E|}{v_{max}} \tag{10}$$

Based on $\bar{t}^s$ and $\bar{t}^e$, we can calculate $\tau(\mathsf{P}, \mathsf{B}_i)$ by Eq. (5). After calculating R_{int} and $\tau(\mathsf{P}, \mathsf{B}_i)$, given an uncertain trajectory necklace database D and a trajectory query T_q, we can compute $Sim_{cont}^{\mathsf{P}\cap\mathsf{B}_i}$ and $Sim_{cont}^{\mathsf{P}\cap\mathsf{T}}$ by Eq. (6) and (7), respectively. If $Sim_{cont}^{\mathsf{P}\cap\mathsf{T}} \geq \alpha$, the result should be returned. We repeat the above mentioned process untill all the necklaces have been visited.

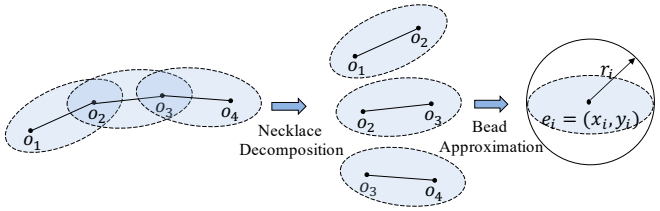


Figure 4: Decomposition of necklaces into circles

3 THE INDEXING STRUCTURE FOR *CSQ*

M-tree [4] is a popular indexing structure for efficient range query and k-nearest neighbor query. But we cannot directly adopt it in this work, because M-tree cannot handle entries like ellipse or beads. Therefore, based on M-tree, we propose a novel indexing structure called Uncertain Trajectory M-tree (UTM-tree), which is discussed next.

3.1 Decomposing the Necklace

First, we decompose the necklaces into beads. After decomposition, we use a circle to approximate each bead, and the circle is centered at the center of the bead (x, y) with radius $r = a/2$ (see in Fig. 4). Each circle is considered an individual entry and is stored independently in the leaf node of the UTM-tree.

Algorithm 1: The Insertion Algorithm: Insert(E_i, N_j)

Input : A new entry $E_i = [e_i, r_i, t_i, t_{i+1}, \texttt{TID}, \texttt{Ptr}(E_p), \texttt{Ptr}(E_n)]$, a tree node $N_j = [E_j, R_j, t_{min}, t_{max}, \texttt{ptr}]$
Output: A UTM-tree with E_i added

```
if N_j is not a leaf node then
    for all the child nodes N_j^sub of N_j do
        Let R_j^sub denote the covering radius of N_j^sub, calculate d(E_i, N_j^sub);
        if ∃ at least one N_j^sub s.t. d(E_i, N_j^sub) + r_i ≤ R_j^sub then
            Insert(E_i, N_j^sub) where d(E_i, N_j^sub) is minimun;
        else
            Insert(E_i, N_j^sub) where d(E_i, N_j^sub) + r_i − R_j^sub is minimum;
else
    if N_j is not full then
        Add E_i into N_j, update t_min and t_max;
        if d(E_i, N_j) + r_i > R_j then R_j = d(E_i, N_j) + r_i ;
    else Split(E_i, N_j) ;
return An updated UTM-tree after inserting E_i into N_j
```

3.2 Building the UTM-tree

Next, we describe how to construct a UTM-tree, including the format of an entry/node, entry insertion, splitting policy, selection of routing objects, and query processing. An illustration of the UTM-tree is shown in Fig. 5.

3.2.1 Format of the entry. Same as the M-tree, in a UTM-tree, all the entries are stored in the leaf nodes. Each node in the tree can store at most M entries, which is also called the capacity of the tree. Specifically, we consider each circle as the entry of the UTM-tree, and each entry is in format of E_i. Here, $E_i = [e_i, r_i, t_i, t_{i+1}, \texttt{TID}, \texttt{Ptr}(E_p), \texttt{Ptr}(E_n)]$, where $e_i = (x_i, y_i)$ and r_i are the center and radius of the approximating circle, t_i and t_{i+1} are the observation timestamps from the original trajectory that form the bead, TID is the trajectory identifier which the bead belongs to, and $\texttt{Ptr}(E_p)$ and $\texttt{Ptr}(E_n)$ are doubly linked list pointers for the previous/next entry that corresponding to beads in the original trajectory.

In the UTM-tree, each node selects an entry from the leaf node as its routing object (similar to M-tree). The format of a node is represented as $N_j = [E_j, R_j, t_{min}, t_{max}, \texttt{ptr}(N_j^{sub})]$, where E_j is its routing object, R_j is the covering radius that covers all the entries stored in N_j. t_{min} and t_{max} are the minimum and maximum time for all the entries stored in N_j, and ptr is a pointer pointing to its child node N_j^{sub} if N_j is a non-leaf node.

3.2.2 Insert an Entry. First, we define the distance between an entry E_i and a node N_j as $d(E_i, N_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where (x_i, y_i) is the center of E_i, and (x_j, y_j) is the center of the rounting object of N_j. The insertion algorithm recursively descends the tree to locate the most suitable leaf node for E_i. We first find the leaf node such that after adding E_i, no enlargement of the covering radius is needed, i.e., $d(E_i, N_j) + r_i \le R_j$. If more than one child nodes are found, then the node that is closest to E_i will be chosen. However, if no such node exists, the choice is to minimize the increase of the covering radius, i.e., $d(E_i, N_j)+r_i-R_j$ is minimum. After insertion, if $d(E_i, N_j)+r_i > R_j$, then we update R_j as $R_j = d(E_i, N_j)+r_i$. Besides, t_{min} and t_{max} for all the visited nodes should be updated as well. After inserting all the entries into the UTM-tree, we use a doubly linked list to connect those entries from the same trajectory (see in Fig. 5). The details of the insertion algorithm is shown in Algorithm 1.

Algorithm 2: The Split Algorithm: Split(E_i, N_j)

Input : An entry $E_i = [e_i, r_i, t_i, t_{i+1}, \texttt{TID}, \texttt{Ptr}(E_p), \texttt{Ptr}(E_n)]$, a leaf node $N_j = [E_j, R_j, t_{min}, t_{max}, \texttt{ptr}]$
Output: A UTM-tree that splits N_j into N_j^1 and N_j^2

```
Let N_j be the set of all N_j's entries plus E_i;
Select two routing objects E_j^1 and E_j^2 from N_j based on m_RAD algorithm [4], and then partition N_j into two sets, N_j^1 and N_j^2;
Allocate a new node N_j', store N_j^1 in N_j and N_j^2 in N_j';
Replace N_j's rounting object E_j with E_j^1, update N_j's R_j;
Set E_j^2 as N_j''s rounting object, and update N_j''s R_j';
if N_j is not the root of the tree (N_j^par is N_j's parent) then
    if N_j^par is full then Split(N_j', N_j^par) ;
    else add N_j' into N_j^par;
else Allocate a new root node N_r, set N_r as parent node for N_j and N_j' ;
Update t_min and t_max for all the visited nodes;
return An updated UTM-tree after spliting N_j by E_i
```

3.2.3 Split a Node. E_i cannot be inserted into a leaf node N_j if there are M entries in N_j. We need to split N_j into two nodes. Particularly, let $\mathcal{N}_j$ denote the set of all N_j's entries plus E_i. We find two rounting objects E_j^1 and E_j^2 from $\mathcal{N}_j$ and partition N_j into two nodes N_j^1 and N_j^2 based on the *m_RAD* algorithm [4]. After partitioning, the sum of the covering radius $R_j^1 + R_j^2$ is miminum. The split algorithm is shown in Algorithm 2.

3.2.4 Data Filtering for CSQ Query. In this part, we discuss how to filter qualified candidates for *CSQ* using UTM-tree.

Given a root node N_r and a query trajectory $\mathsf{T}_q = \{\mathsf{P}_1, \mathsf{P}_2, ..., \mathsf{P}_d\}$ where $\mathsf{P}_k = (x_k^c, y_k^c, r_k^c, t_k^s, t_k^e)$ for $k \in [1, d]$, we can first filter all the entries $E_i = [e_i, r_i, t_i, t_{i+1}, \texttt{TID}, \texttt{Ptr}(E_p), \texttt{Ptr}(E_n)]$ such that (1) $d(e_i, (x_k^c, y_k^c)) \le r_i + r_k^c$ and (2) $t_i \le t_k^s$ and $t_{i+1} \ge t_k^c$ for all pairs of (N_r, P_k). Next, based on the filtered entries, we can get a list of trajectory identifiers TID (s). For each trajectory T from this list, we can calculate $Sim_{cont}^{\mathsf{P}_k \cap \mathsf{T}}$, and then check if $Sim_{cont}^{\mathsf{P}_k \cap \mathsf{T}} \ge \alpha$ or not. The UTM-tree indexing structure can significantly reduce

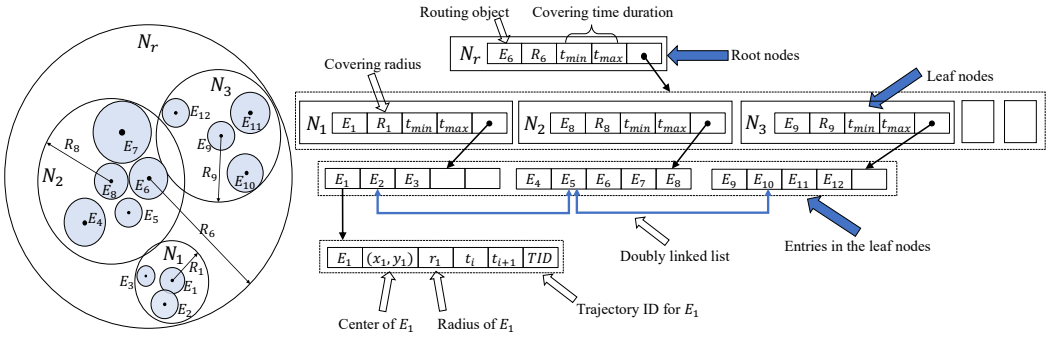


Figure 5: An example of the UTM-tree with capacity of 5

the number of trajectories to be visited, thus it can greatly increase the performance of the *CSQ* query. The details of the data filtering algorithm is shown in Algorithm 3.

Algorithm 3: The filtering algorithm: Filter(N_r, P_k)

Input : A root node N_r, and P_k where $\mathsf{P}_k = (x_k^c, y_k^c, r_k^c, t_k^s, t_k^e)$
Output: A list of the trajectories IDs that satisfy the query

if N_r *is not a leaf node* **then**
  for *all the children nodes* N_r^{sub} *in* N_r *where* $N_r^{sub} = [E_r^{sub}, R_r^{sub}, t_{min}^{sub}, t_{max}^{sub}, \mathtt{ptr}(N_r^{sub(sub)})]$ **do**
    if $(t_{min}^{sub}, t_{max}^{sub}) \cap (t_k^s, t_k^e) \neq \emptyset$ **then**
      if $d(E_r^{sub}, (x_k^c, y_k^c)) \leq r_k^c + R_r^{sub}$ **then** Filter(N_r^{sub}, P_k);
      else Prune N_r^{sub};
    else Prune N_r^{sub};
else
  for *every* $E_i = [e_i, r_i, t_i, t_{i+1}, \mathtt{TID}, \mathtt{Ptr}(E_p), \mathtt{Ptr}(E_n)]$ *in* N_r **do**
    if $(t_i, t_{i+1}) \cap (t_k^s, t_k^e) \neq \emptyset$ **then**
      if $d(e_i, (x_k^c, y_k^c)) \leq r_k^c + r_i$ **then** Add TID into an ID list;
      else Prune E_i;
    else Prune E_i;
return *The ID list*

4 EXPERIMENTAL EVALUATION

We implemented the systems with Python on an Intel(R) Core(TM) i7-6700 CPU @3.60GHz running Windows 64-bit OS with 32 GB RAM. The details of experimental settings and the performance evaluation are discussed as follows.

4.1 Settings

In this part, we discuss the datasets, experimental settings, and the basic approaches in this work.

• Beijing Taxi Dataset: This dataset [12] contains the GPS trajectories of 10357 taxis from Feb. 2 to Feb. 8, 2008 in Beijing. The total number of points in this dataset is about 15 million, and the total distance traversed by the trajectories is 9 million kilometers. The average sampling interval is 177 seconds, with an average length of 623 meters.

Table 2: The summary of simulation settings

Parameters	Settings
# of observations in one trajectory (Num_o)	30
# of trajectories (Num_T)	25K, 50K, 100K, 250K, 500K
Radius of each virtual landmark (r_k^c)	5m, 10m, 15m, 20m, 25m
# of virtual landmark in a query (Num_P)	20, 25, 30, 35, 40
Query time interval (τ_q)	1h, 2h, 4h, 6h, 8h

• Experimental Settings: There are two metrics for evaluating the performance of the proposed UTM-tree, they are (1) running time and (2) the number of visited trajectories. Table 2 shows all the parameter settings in the experiment. Specifically, the number of observations in each trajectory is set to 30. For a virtual landmark $\mathsf{P}_k = (x_k^c, y_k^c, r_k^c, t_k^s, t_k^e) \in T_q$, query time interval equals $\tau_q = t_k^e - t_k^s$. Moreover, considering a day with 24 hours, τ_q is randomly selected from 8:00 am to 6:00 pm in the same day. For each P_k, its latitude coordinate x_k^c and longitude coordinate y_k^c are randomly selected in the city of Beijing such that $x_k^c \in (116.1650, 116.6201)$ and $y_k^c \in (39.7133, 40, 0070)$. Besides, the maximum speed v_{max} in-between any two continuous observations is set to a random number in $[10\ km/h, 50\ km/h]$.

• Other approaches evaluated: To evaluate the performance of the UTM-tree, we implemented two other approaches for comparison purposes, they are (1) baseline method and (2) temporal-first matching (TF-matching). In the baseline method, we do not index the trajectories at all. More specifically, we evaluate every pair of $(\mathsf{B}_i, \mathsf{P}_k)$ for the *CSQ* query. In the TF-matching method, we follow the steps in [9] to build a temporal filtering tree. Specifically, we split a day into 12 time slots by considering every two hours as a time slot. And then, we store each trajectory into the corresponding nodes. In the TF-matching method, all the trajectories are processed by the temporal filtering tree.

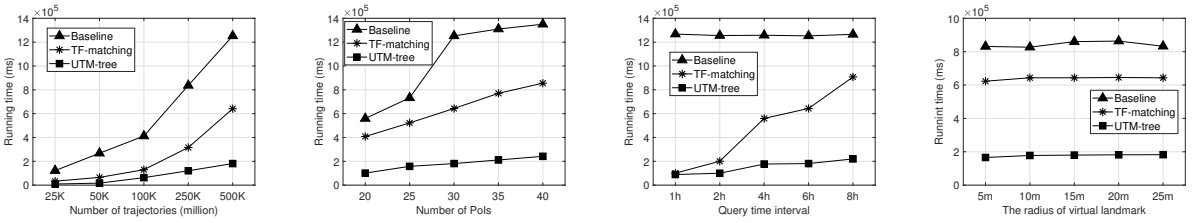
4.2 Performance Evaluation

This section compares the proposed UTM-tree with the baseline and the TF-matching in terms of query running time and the number of visited trajectories. The number of visited trajectories means how many trajectories from the original database still need to be considered after using the UTM-tree for spatial filtering and temporal filtering.

First of all, for the three approaches, the query running time (in Fig. 6 (a)) and the number of visited trajectories (in Fig. 7 (a)) are increasing as Num_T increase. Second, we can see that the UTM-tree is much faster than the baseline method and the TF-matching method. Third, we found that for the UTM-tree, as Num_T becomes larger, the increase for the query running time and the number of visited trajectories is not significant.

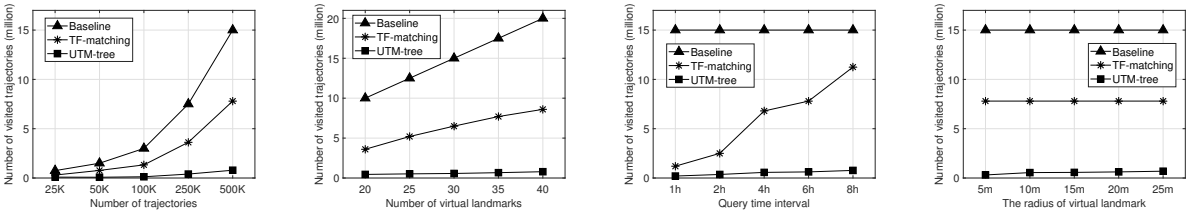
Next, the comparison results for three approaches under the different Num_P are shown in both Fig. 6 (b) and Fig. 7 (b). We can see that Num_P has the least influence on the performance of the UTM-tree. In contrast, the performance of the baseline method and the TF-matching approach is degraded dramatically.

The influence of τ_q on the three approaches under different experimental settings are shown in both Fig. 6 (c) and 7 (c). The running time and the number of visited trajectories in TF-matching



(a) Query running time vs. $\mathsf{Num_T}$ (b) Query running time vs. $\mathsf{Num_P}$ (c) Query running time vs. τ_q (d) Query running time vs. r_n^c

Figure 6: The comparison of baseline method, TF-matching, and UTM-tree for query running time



(a) # of visited trajectories vs. $\mathsf{Num_T}$ (b) # of visited trajectories vs. $\mathsf{Num_P}$ (c) # of visited trajectories vs. τ_q (d) # of visited trajectories vs. r_n^c

Figure 7: The comparison of baseline method, TF-matching, and UTM-tree for number of visited trajectories

approach increase significantly when τ_q becomes wider. However, the UTM-tree is more stable against the changes in τ_q.

Last, Fig. 6 (d) and 7 (d) compare the performance of the three approaches under different r_k^c. Both the three approaches do not change significantly with different r_k^c. Overall speaking, the UTM-tree is more efficient and stable than the baseline method and the TF-matching approach in terms of running time and the number of visited trajectories under different experiment settings. Specifically, the UTM-tree runs faster than other approaches and is less sensitive to query time intervals, virtual landmark radius, and the number of virtual landmarks in each query.

5 RELATED WORK

(1) *Indexing uncertain trajectories.* There are a number of indexing structures, which have been proposed for uncertain spatio-temporal data, such as UTH [13], Grid-based indexing [3], and UST-tree [5]. However, UTH-tree was used for indexing uncertain trajectories on the road network. UST-tree was used for approximating diamond-based moving objects that follow the Markov-chain model. Grid-based approach was not efficient for indexing beads and necklaces since it is time-consuming to compute the overlapped region between grids and ellipses. Therefore, we devise a novel indexing structure called UTM-tree, which is based on the classic M-tree and is efficient for indexing the uncertain trajectories in the form of beads.

(2) *Queries for uncertain trajectories.* There are many studies that were proposed for querying uncertain trajectories, such as spatio-temporal similarity join [9], semantic similarity join [3], nearest-neighbor queries [11], and top-k similarity query [6]. However, most of them retrieved qualified trajectories based on some spatial/temporal criteria. None of the previous works study the problem of contact similarity in terms of spatial intersection and longest contact time duration among trajectories. This work is the first research to formally define the problem of contact similarity and propose the corresponding *CSQ* for the problem.

6 CONCLUSION

In this work, we have formally defined the concept of contact similarity and proposed a novel query, called *CSQ*. Next, we designed a novel indexing structure called UTM-tree, for managing and querying uncertain trajectories. Besides, we conducted extensive experiments on the Beijing Taxi dataset. The experimental results demonstrated the efficiency and stability of the UTM-tree on *CSQ*. There are many interesting future directions, e.g., (1) contact modeling between MOs on road networks, (2) indoor contact modeling between MOs, (3) *CSQ* in terms of contact frequency, (4) second-generation contact between MOs, and (5) performance evaluation with more real-world datasets.

REFERENCES

[1] Prithu Banerjee, Sayan Ranu, and Sriram Raghavan. 2014. Inferring uncertain trajectories from partial observations. In *2014 IEEE ICDM*. IEEE, 30–39.

[2] Ionut Cardei, Cong Liu, Jie Wu, and Quan Yuan. 2008. DTN routing with probabilistic trajectory prediction. In *WASA*. Springer, 40–51.

[3] Lisi Chen, Shuo Shang, Christian S Jensen, Bin Yao, and Panos Kalnis. 2020. Parallel Semantic Trajectory Similarity Join. In *2020 IEEE 36th ICDE*. 997–1008.

[4] Paolo Ciaccia, Marco Patella, Fausto Rabitti, and Pavel Zezula. 1997. Indexing metric spaces with m-tree. In *SEBD*, Vol. 97. 67–86.

[5] Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Züfle. 2012. Indexing uncertain spatio-temporal data. In *Proceedings of the 21st ACM CIKM*. 395–404.

[6] C. Ma, H. Lu, L. Shou, and G. Chen. 2013. KSQ: Top-k Similarity Query on Uncertain Trajectories. *IEEE TKDE* 25, 9 (2013), 2049–2062.

[7] Mark EJ Newman. 2002. Spread of epidemic disease on networks. *Physical review E* 66, 1 (2002), 016128.

[8] Dieter Pfoser and Christian S Jensen. 1999. Capturing the uncertainty of moving-object representations. In *SSD*. Springer, 111–131.

[9] Shuo Shang, Lisi Chen, Zhewei Wei, Kai Zheng, and Panos Kalnis. 2017. Trajectory similarity join in spatial networks. *VLDB Endowment* (2017).

[10] Goce Trajcevski, Alok Choudhary, Ouri Wolfson, Li Ye, and Gang Li. 2010. Uncertain range queries for necklaces. In *2010 11th MDM*. IEEE, 199–208.

[11] Goce Trajcevski, Roberto Tamassia, Hui Ding, Peter Scheuermann, and Isabel F Cruz. 2009. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *Proceedings of the 12th EDBT*. 874–885.

[12] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD*. 316–324.

[13] Kai Zheng, Goce Trajcevski, Xiaofang Zhou, and Peter Scheuermann. 2011. Probabilistic range queries for uncertain trajectories on road networks. In *Proceedings of the 14th EDBT*. 283–294.