A System for Discovering Regions of Interest from Trajectory Data

Muhammad Reaz Uddin, Chinya Ravishankar, and Vassilis J. Tsotras

University of California, Riverside, CA, USA
{uddinm,ravi,tsotras}@cs.ucr.edu

Abstract. We show how to find regions of interest (ROIs) in trajectory databases. ROIs are regions where a large number of moving objects remain for at least a given time interval. Our implementation allows a user to quickly identify ROIs under different parametric definitions without scanning the whole database. We generalize ROIs to be regions of arbitrary shape of some predefined density. We also demonstrate that our methods give meaningful output.

Keywords: Spatio-temporal database, Trajectory, Region of Interest.

1 Introduction

The widespread use of GPS-enabled devices has enabled many applications that generate and maintain data in the form of *trajectories*. Novel applications allow users to manage, store, and share trajectories in the form of GPS logs, and find travel routes, interesting places, or other people interested in similar activities.

This demonstration is based on the paper [1], where we give a novel and more intuitive definition of ROIs and propose a framework for identifying them. Recent works on discovering ROIs from trajectory data [2] define ROI as an (x, y) average of the points of a subtrajectory in which the object moves less than a prespecified distance threshold δ and takes longer than a prespecified time threshold τ. If either δ or τ changes, the entire trajectory database must be re-scanned. In contrast, our work removes this important limitation.

It is more intuitive to define ROIs in terms of speed. If an object takes at least time τ to travel at most distance δ, it maintains an average speed no more than $\frac{\delta}{\tau}$ for at least time τ. In our framework, we actually use a speed *range* to define ROIs, as this leads to a more generic definition. Further, we introduce the notion of *trajectory density* to define ROIs. In summary, our ROI definition uses (1) a range of speed that an object maintains while in an ROI (2) a minimum duration of staying in an ROI area and (3) the density of objects in that area.

We build an index on object speeds to avoid scanning the whole database. Given a range or a particular speed, we first retrieve trajectory segments with that speed using this index. We then verify the minimum stay duration condition. Objects that fulfill the speed and duration condition are *candidate objects*. Finally, we identify dense regions of candidate objects.

2 Defining Regions of Interest

Conceptually, an ROI is intended to be a region where moving objects pause or wait or move slow in order to complete activities that are difficult or impossible to carry out while in fast motion. Examples of ROIs are restaurants, museums, parks, places of work, and so on. Generally, individual trajectories display idiosyncrasies, so ROIs are best defined in terms of collective behaviors of a collection of trajectories. That is, a collection of trajectories is needed to identify a location as an ROI.

The duration of an object's stay in a location is important in filtering out spurious ROIs, e.g. busy road intersections. So we will require a *minimum stay duration* for objects at ROIs. Nevertheless, if an object spends a long time in a large spatial region, a city, say, then that large region should not be considered as an ROI either. Hence, we must also consider the geographic extent of the object's movement, that is, the maximum area within which an object remains (or the maximum distance traveled by an object) during the minimum stay duration. Finally, to capture the collective behavior we consider the density of candidate objects in such a region. We identify dense regions adapting the *point-wise dense region* approach of [3].

Definition 1. *A region R is a* ***region of interest*** *if every point $p \in R$ has an l-square neighborhood containing segments from at least N distinct trajectories with object speeds in the range $[s_1, s_2]$, and where each such object remains in R for at least time τ before leaving R. The parameters l, N, τ, s_1, s_2 are user-defined.*

3 Indexing Trajectory Segments by Speed

Typically, objects in an ROI will maintain very low (or zero) speed. Hence, if we can quickly retrieve and analyze low speed trajectory segments, we can reduce query costs significantly.

Let $s_{\max}$ and $s_{\min}$ be the maximum and minimum speeds specifiable in an ROI query. We partition the speed values into *index ranges* $\mathcal{R} = [s_{\min}, s_1)$, $[s_1, s_2)$, $\ldots$, $[s_{n-1}, s_{\max})$. These ranges can be of arbitrary length. We maintain one bucket for each index range, with bucket B_i holding trajectory segments with speed range $[s_i, s_{i+1})$.

We consider the segments of a trajectory sequentially, and compute speeds assuming linear motion between two successive timestamps. If a series of consecutive segments fall within the same speed range, we combine them into one subtrajectory, and insert it into the index as one entry. Thus each entry in an index bucket points to a subtrajectory all of whose segments fall into within the speed range of the bucket.

We assume trajectories are sorted according to TID, so that subtrajectories in the buckets are also sorted according to TID. Having TID sorted entries in the buckets allows to perform a merge join to reconstruct trajectories from these buckets. When new trajectories are added to the database the index can easily be updated using the above algorithm.

4 Finding Regions of Interest

We find ROIs in three steps. First, we retrieve the appropriate buckets from the index. In the second step, we collect subtrajectories spanning multiple buckets by performing a merge-join, and check the stay durations. In the third step, we find regions with line segment density N/l^2, where each of N segments has to be from different trajectories.

It is straightforward to retrieve the segments falling into a given speed range $[s_1, s_2)$ using the speed index. No further discussion is needed.

4.1 Step 2: Verifying the Duration Condition

In this step, we consider only the buckets obtained from the previous step. To verify the duration condition for each trajectory we must join subtrajectories with same TID from different buckets. Let the query speed range include buckets B_i and B_j, and let $S_i \in B_i$ and $S_j \in B_j$ be subtrajectories. Let the start and end timestamps for S_i and S_j be $[t_{i1}, t_{i2}]$ and $[t_{j1}, t_{j2}]$ respectively. If S_i and S_j have the same TID and $t_{i2} = t_{j1}$ or $t_{i1} = t_{j2}$, then S_i and S_j should be merged into a single subtrajectory. The object's stay duration is the interval between the first and the last timestamps of the merged subtrajectory. We discard all subtrajectories with stay duration less than τ after merge, since they do not fulfil the stay duration condition.

In addition to minimum stay duration, our implementation also supports other temporal conditions, such as time intervals and weekdays/weekends. For example, ROIs during any weekday with $\tau = 15$ to 30 minutes, carry different semantics than those found in the afternoon or evening of any weekend, with a few hours of stay duration.

4.2 Step 3: Finding Dense Regions

This step involves finding points p whose l^2-neighborhood contains at least N distinct trajectories. For our purpose we use the Pointwise Dense Region (PDR) method [3] which was originally presented for point objects. The work in [3] describes two variations: (1) an exact, and (2) an approximate method. We use both of them. [3] uses Chebyshev polynomials to approximate the density of 2D points. We take the middle point p_m of each trajectory segment, and update the Chebyshev coefficient for the l-square neighborhood of p_m. A trajectory segment is a straight line between two points of a trajectory recorded at consecutive timestamps.

5 Demonstration

We develop a user interface where a user can specify values of query parameters e.g., speed, stay duration, other temporal conditions etc. Users can also select their data or use datasets on which we test our implementation. We show the

Table 1. Description of real data set

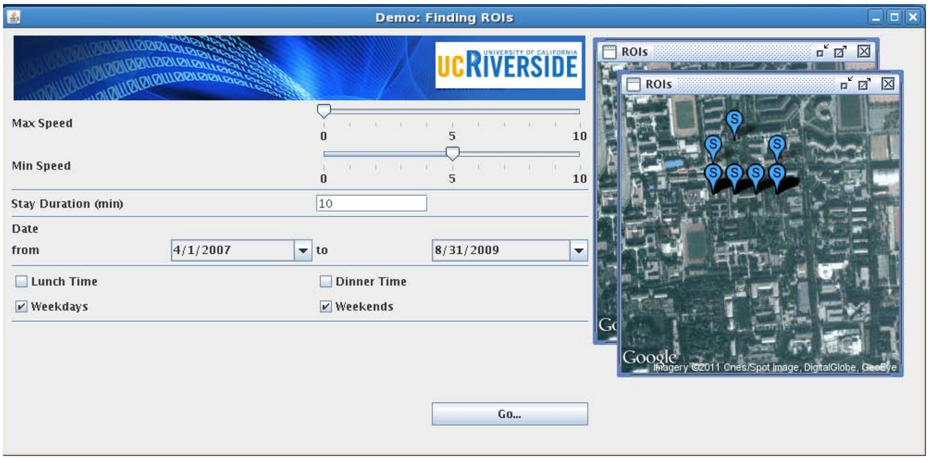
Description	Time of collection	♯trajectories
GeoLife Data: Beijing, China [4].	Apr 2007 to Aug 2009	165
TaxiCab Data: San Francisco, USA [5].	2008-05-17 to 2008-06-10	536

ROIs found by our methods using Google Maps API. The user can change any parameter value leaving others same and see the change. For example after identifying all ROIs for a region user can select only weekends or weekdays ROIs and see the difference. Figure 1 shows the user interface of our system.

Table 1 provides the description of the real datasets that we use to test our implementation. Using a short stay duration (15 to 30 min) for GeoLife data, we found bus stops, railway and subway stations, the Tsinghua University canteen, etc. We then considered weekends and a longer stay duration (1.5 to 4 hr). This resulted in ROIs in (1) the Sanlitun area which houses many malls, bars and is a very popular place, (2) the Wenhua square which contains churches, theaters, and other entertainment places, and (3) Zhongguancun, referred to as 'China's Silicon Valley', having a lot of IT and electronics markets. Figure

Figure 2(a) shows all the ROIs found using the TaxiCab dataset. We further zoomed in to ROIs and found (b) The San Francisco international airport, (c) a car rental, (d) the main downtown, union square, (e) San Francisco Caltrain station (f) the yellow cab access road. We also found hotels e.g. Star Wood, Westin, Mariott, Radisson, Ramada Plaza, Regency hotel, etc. These were found for short stay duration of 10 minutes. When the stay duration was increased to 12 hours we found only yellow cab access road, while for 2 − 3 hours of stay duration we also found the airport.

Figure 2(g) and (h) shows Sanlitun and Zhongguancun area respectively in Beijing. When considering lunch and dinner time we found places that contain



Max Speed 0 5 10
Min Speed 0 5 10
Stay Duration (min) 10
Date
from 4/1/2007 to 8/31/2009
☐ Lunch Time ☐ Dinner Time
☑ Weekdays ☑ Weekends

Fig. 1. The User Interface

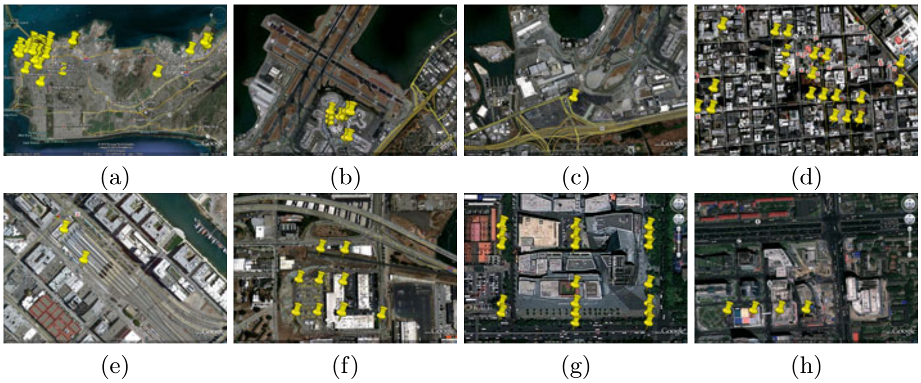


Fig. 2. ROIs identified for the TaxiCab and GeoLife data

many restaurants. Interestingly ROIs found at lunch time contain regions near the Microsoft China head quarters which are absent in dinner time ROIs. Finally, we identified ROIs on each individual day from April 2007 to August 2009. These resulted in (1) the Olympic media village, the Olympic sports center stadium during the Olympics 2008, (2) Peking University when the 'Regional Windows Core Workshop 2009 - Microsoft Research' was taking place in the PKU campus, (3) areas near the Great Wall in a weekend, (4) the Beijing botanical gardens, (5) the Celebrity International Grand Hotel, Beijing, etc.

References

1. Uddin, R., Ravishankar, C., Tsotras, V.J.: Finding regions of interest from trajectory data. In: MDM (to appear, 2011)
2. Cao, X., Cong, G., Jensen, C.S.: Mining significant semantic locations from gps trajectory. In: VLDB, pp. 1009–1020 (2010)
3. Ni, J., Ravishankar, C.V.: Pointwise-dense region queries in spatio-temporal databases. In: IEEE ICDE, pp. 1066–1075 (2007)
4. `http://research.microsoft.com/en-us/projects/geolife/`
5. `http://crawdad.cs.dartmouth.edu`