2013 IEEE 13th International Conference on Data Mining Workshops

A Novel Approach to Trajectory Analysis Using String Matching and Clustering

Madhuri Debnath
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: madhuri.debnath@mavs.uta.edu

Praveen Kumar Tripathi
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: praveen.tripathi@mavs.uta.edu

Ramez Elmasri
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: elmasri@cse.uta.edu

***Abstract*—Clustering of sub-trajectories is a very useful method to extract important information from vast amounts of trajectory data. Existing trajectory clustering algorithms have focused on geometric properties and spatial features of trajectories and sub-trajectories. In contrast to the existing trajectory clustering algorithms, we propose a new framework to cluster sub-trajectories based on a combination of their spatial and non-spatial features. This algorithm combines techniques from grid based approaches, spatial geometry and string processing. First, we convert each trajectory into a representative sequence that captures the trajectory direction and location. We identify common sub-trajectories from all the sequences using a modified string matching algorithm. Then, we extract non-spatial features from the common sub-trajectories. Finally, we present a density based clustering algorithm to cluster the sub-trajectories. Experimental results show that our framework correctly discovers groups of similar sub-trajectories with their similar non-spatial features.**

I. Introduction

Because of vast improvements in GPS and current sensor technologies, large scale trajectory data are available. The data provided by the technologies are raw data. Hence it becomes significant to discover important and meaningful information by analysing them. Some examples of trajectory data are vehicle position data, hurricane track data, animal or pedestrian movement tracking data, radio frequency identification (RFID) data.

Trajectories are represented as a sequence of spatio-temporal points. Analysing trajectory data has many other applications such as managing the traffic pattern of vehicles, monitoring and predicting weather conditions, examining wild animal behaviour and movement, as well as analysing the spread of disease. A number of attempts have been made in this domain to analyse these kind of data sets. Some of these analyses can be found in [1],[2],[3],[4],[5].

Existing trajectory clustering algorithms have focused on spatial proximity and spatial features (latitude and longitude) of trajectories. Similarity of non-spatial attributes of trajectories has not been considered. Examples of non-spatial attributes for hurricane trajectories are wind speed, length, area coverage, whereas for that of vehicle trajectories are speed, length and frequency of stops between movements. These attributes can be significantly different from each other. For example, two hurricane trajectories or their sub-trajectories may have common tracking patterns based on spatial location and direction, but they may have different wind speed and wind pressure or time-span of two hurricane can be different.

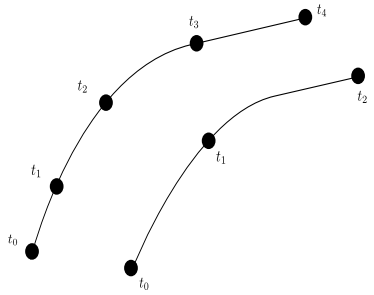

Fig. 1. An example of two trajectories

Example: Consider the two trajectories in Fig. 1. The trajectory data that we use is from the hurricane dataset [6]. In this dataset, time points in a trajectory are 6 hours apart. Trajectory that starts at t_0 and ends at t_4 has total duration of 24 hours. Another trajectory starting at t_0 and ending at t_2 has total duration of 12 hours. It is obvious that average wind speed of the second trajectory is higher than the first one, because they have the same approximate length but a different number of points.

In this paper, we aim to cluster trajectories considering both spatial features and non-spatial features. The main motivation behind this is to find groups of spatial dense regions of trajectories with similar non-spatial attribute behaviour. Here we give one example to illustrate that discovering common sub-trajectories considering both type of feature is useful.

1) Storm trajectory analysis has an important application in forecasting hurricane landfall information [2]. Storms with high wind speed and intensity is more significant than those of low speed and intensity value. So, analysing these attributes will be useful to predict the storm locations with high wind speed and intensity.

The rest of the paper is organized as follows. In Section 2, we review some related works. We describe our proposed algorithm in Section 3. Section 4 presents experimental results of our algorithm. Finally, Section 5 concludes the paper.

978-0-7695-5109-8/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDMW.2013.130

II. Related Works

In this section, we briefly describe some works that are most relevant to this one. In [7], the authors proposed a model based clustering algorithm, where a set of trajectories are represented using a regression mixture model. EM algorithm is used to determine cluster membership. They considered the whole trajectory as a basic unit of clustering.

In [2], the authors proposed a partition and group framework to cluster trajectories. In the partitioning phase, trajectories are divided into some line segments. This division has been done using the notion of characteristic points which reflect the most significant points in the trajectory. In the grouping phase, these line segments are clustered using DBSCAN algorithm [8]. The obvious drawback of this algorithm is that only the line segments are being clustered.

In [1], the authors proposed a modular approach to cluster sub-trajectories. This approach is based on combination of techniques from computational geometry, string processing and data mining.

In [9], the authors proposed a technique for mining a spatio temporal pattern called the flocking behaviour in an online fashion. The flocking patterns refer to the set of the trajectories that remain close to each other for some reasonable time interval. The authors considered both the time information along with the spatial attributes in mining the flocking behaviour.

In [5], the authors proposed a non-parametric approach to cluster spatial trajectories. This article deals with trajectory clustering and the post analysis of the clustering results. In this approach, they proposed a clustering algorithm that uses a randomized hill climbing technique to find some local maxima of the density function. Final clusters are obtained by grouping together the trajectories belonging to the same local maxima. The post processing of the obtained spatial clusters is performed to get more domain specific knowledge.

Another important analysis of spatio-temporal tracking data has been proposed in [10]. The authors proposed a framework for mining the sequential patterns from the spatio-temporal data. They proposed a sequence index that is important in identifying the significant spatio-temporal sequential patterns from the spurious ones. A novel algorithm called Slicing-STS-Miner has been proposed to use the given sequence index in order to efficiently obtain the spatio-temporal sequential patterns.

In [11], the authors extended their previous work on trajectory clustering [2] and proposed a new algorithm of trajectory classification. They proposed two types of clustering: 1) region-based clustering and 2) trajectory based clustering. The motivation is to arrive at the discriminative features, which are very vital in generating the classifier model for the classification task. The first level of the clustering, which is the region level, identifies the higher level, region based features of the trajectories, ignoring the movement based features at this stage. The second level of the clustering identifies the lower level movement based features. These two clustering collaboratively identify the high-quality features for the classification task.

In [12], authors proposed an algorithm for automatically identifying dominant patterns from a dense crowded scenes. They used longest common subsequence algorithm to find similarity of two point track segments and then clustered them to identify smooth dominant motions in a crowded scene.

In [13], authors extended DBSCAN algorithm to incorporate the non-spatial attribute along with spatial attribute, but they did not apply it in trajectory data.

III. Proposed Framework

Our proposed framework has four phases which are, $1)$ dimension reduction, $2)$ trajectory segmentation, $3)$ non-spatial feature extraction and $4)$ clustering.

In the first phase, we map each trajectory from high dimensional (usually 2 or 3) space to one-dimensional space. This mapping simplifies trajectory representation and their comparison in later stages of our framework.

The second phase deals with the identification of common sub-trajectories among the dataset. This phase exploits the string matching concept to identify common sub-trajectories among all trajectories. In this work we use a modified version of Longest Common String Matching (LCS) algorithm [14].

In the third phase, we extract non-spatial features from the sub-trajectories obtained from the second phase. Some examples of non-spatial features are wind speed, trajectory length etc.

In the fourth phase, we cluster the sub-trajectories based on the combination of their spatial and non-spatial features. We use DBSCAN algorithm for clustering.

A. Representation of Trajectory

Each trajectory is represented as a sequence of n spatial locations with time information viz., $(t_0, l_0), (t_1, l_1), (t_2, l_2), (t_3, l_3), ...(t_n, l_n)$. Here, n is the trajectory length. Each location l_i is a 2-dimensional point. The length of one trajectory can be different from another one. At the same time, the shape and movement of each trajectory is different. We can consider one trajectory as $n \times 3$ dimensional matrix, where n = number of points in the trajectory.

$$\text{Trajectory } \mathrm{T}_1 = \begin{bmatrix} t_0 & x_0 & y_0 \\ t_1 & x_1 & y_1 \\ t_2 & x_2 & y_2 \\ .. & .. & .. \\ t_n & x_n & y_n \end{bmatrix}$$

In this phase of our algorithm, we focus on mapping a trajectory from two dimensional space to one dimensional space. Note that, this dimensionality reduction is done for simpler spatial trajectory representation for its better segmentation. There are several techniques to accomplish this [15]. The most prominent ones include the Z-order, Gray Codes and Hilbert Curve. In spatial database management systems, these mapping techniques are used to store and index location information on disk, as disk storage is logically one dimensional device [15]. In this paper, we aim to simplify the trajectory data to facilitate recognition of moving pattern and to compare it with other trajectory data. For this task we propose our novel algorithm.

We divide the whole problem space domain into $M \times N$ grid cells, where each grid cell has equal length and width. We consider each grid cell as a spatial region. Each region is identified with a unique identification number viz., id. The identification number is assigned in row major order. For example, in the first row, $1..N$ column is identified with numbers from 1 to N. In the second row, all columns are identified with $N+1$ to $N+N$ and so on (see **Algorithm 1**).

*1) **Example**:* Let us consider the following example of one trajectory.

$$\text{Trajectory } \mathrm{T}_1 = \begin{bmatrix} 0 & 1.5 & 1.5 \\ 6 & 1.8 & 2.5 \\ 12 & 3.5 & 2.8 \\ 18 & 5.5 & 4.8 \\ 24 & 6.2 & 5.5 \\ 30 & 4.8 & 5.8 \end{bmatrix}$$

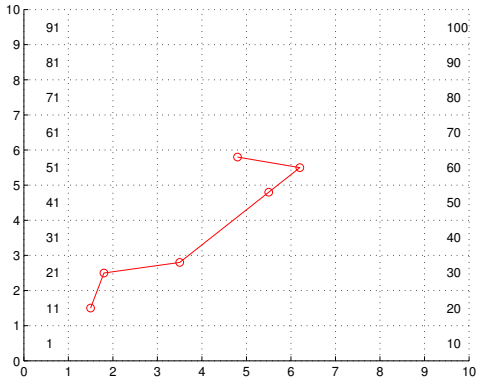

Fig. 2. Trajectory T_1

Let us, divide the problem space domain into 10×10 square grid cells, with each grid cell representing 1 spatial region. Now we assign each grid cell a unique identification number from 1 to 100, as we have a total of 100 grid cells. In first row, columns are identified from 1 to 10, in the second row, they are identified from 11 to 20. In order to map the trajectory T_1 in this grid, it's first point which is viz., $(0, (1.5, 1.5))$ is mapped to grid location number 12. Fig. 2 shows the first and last grid cell number in each row.

We assume that the interval between two successive points is the same (as is given in the dataset) [6]. For example, trajectory represented as T_1 = $(12, 22, 24, 46, 57, 55)$ (see Fig. 2). Hence, a trajectory of length n is a sequence of $T[n]$ = ($l_1, l_2, l_3 ..... l_n$), where l_i is the grid location that approximately represent a trajectory point.

The advantage of this approach is its simplicity and it takes linear time to map each point to an one-dimension space. We can also do the reverse mapping from grid cell representation to approximate 2-D position of a point. For example, if grid id = l_o and we have N columns, then x coordinate can be retrieved as $l_o \mod N$ and y coordinate can be retrieved as $\lceil l_0/N \rceil$ (see **Algorithm 2**). Note that, because of the approximate mapping of the 2-D points to the grid ids the reverse mapping does not guarantee the exact 2-D points, only approximate location.

For example, grid location l_0 = 12 is mapped to 2-D point $(2, 2)$ and l_1 = 22 is converted to 2-D point $(2, 3)$. To measure distance between two grid locations, we measure Euclidean distance between 2-D approximate points of two grid ids (see **Algorithm 3**).

Algorithm 1 Transformation of 2-D point to 1-D value

1: **procedure** ONE-D-TRANSFORM(x, y, N)
2: $p \leftarrow ceil(x + floor(y) * N)$
3: **return** p
4: **end procedure**

Algorithm 2 Transformation of 1-D value to approximate 2-D point

1: **procedure** TWO-D-TRANSFORM(l_0, N)
2: $x \leftarrow l_0 \mod N$
3: $y \leftarrow \lceil l_0/N \rceil$
4: **return** (x, y)
5: **end procedure**

Algorithm 3 Grid distance between two grid location

1: **procedure** GRID-DISTANCE(l_i, l_j, N)
2: $< x_i, y_i >\leftarrow$ Two-D-Transform (l_i,N)
3: $< x_j, y_j >\leftarrow$ Two-D-Transform (l_j,N)
4: $d \leftarrow \sqrt{(x_i^2 - x_j^2) + (y_i^2 - y_j^2)}$
5: **return** d
6: **end procedure**

B. Segmentation Algorithm

In the previous section, we focused on the approximate and simplified representation of a trajectory. In this section, we define an approach to segment a trajectory into sub-trajectories. We compare two trajectories and find the approximate common segments between them. This idea originates from Longest Common Substring (LCS) matching algorithm [14].

1) Longest Common Approximate Trajectory Segments (LCATS): Given two trajectory sequence S of length m (s_1,s_2,s_3,.....s_m) and T of length n (t_1,t_2,t_3,.....,t_n). Let X (x_1,x_2,...,x_p) be a sub-sequence of S and Y (y_1,y_2,..y_p) be a sub-sequence of T. X and Y are called approximate common trajectory segments of S and T if for each points of X and Y, grid-distance(x_i,y_i) $\leq \epsilon$. In LCATS Problem, we wish to find the longest common trajectory segments between two trajectories. We have used dynamic programming approach to find the LCATS between two trajectories.

Example: Consider the following example (see Fig. 3). Here, S, T, V are three different trajectories. (a, b, c, d) and (h, i, j, k) are LCATS of trajectory S and T. (m, n, o) and (p, q, r) are LCATS of trajectory S and V.

2) Identifying sub-trajectories : We consider all trajectories to find the common patterns with other trajectories. To reduce the cost of comparing a trajectory sequence with another trajectory sequence, we come up with the idea to use the notion

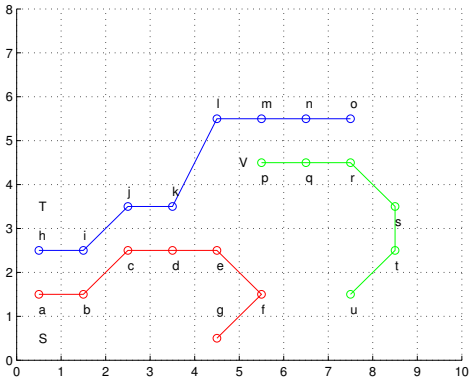

Fig. 3. Longest Common Approximate Sub-trajectories

of Minimum Bound Rectangle (MBR). MBR is a popular technique for indexing spatial objects. It is also used to indicate approximate spatial positions of spatial objects.

3) ***Example****:* Consider the example in Fig. 4. Minimum bounding rectangle of trajectory 1 and 2 intersects with each other. So, there is a possibility that they have some common segments between them. But MBR of trajectory 3 is not intersecting with any one of them. So, we do not consider trajectory 3 to compare with trajectory 1 and 2.

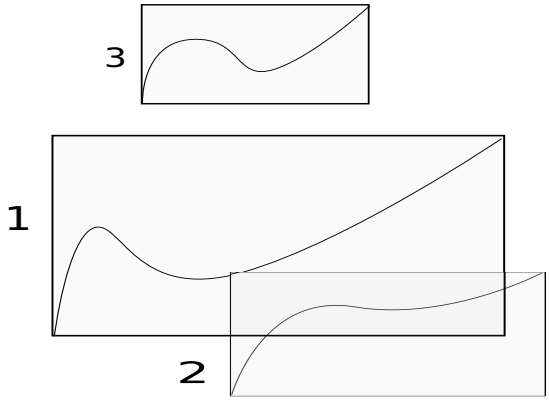

Fig. 4. MBR of three trajectories

C. Non-spatial feature extraction

After the segmentation phase described in the previous section, we obtain a set of sub-trajectories. In this section, we identify some non-spatial features associated with them. Non-spatial features are application dependent. In different application domain, different non-spatial features provide significant information. For example, In storm trajectory data some significant features are:

1) Average wind speed during the storm
2) Time-length of each storm
3) Wind pressure
4) storm intensity
5) Area coverage

Algorithm 4 Identifying Sub-trajectories

Input: T : a set of trajectories $(T_1, T_2, ...T_n)$
Output: S : a set of sub-trajectories $(s_1, s_2, ...s_m)$

```
1:  procedure IDENTIFY-SUB-TRAJECTORIES(T)
2:      S ← ∅
3:      for i = 1 → n do
4:          for j = i + 1 → n do
5:              if MBR of T_i intersects MBR of T_j then
6:                  U ←LCATS(T_i,T_j)
7:                  if U is not Null then
8:                      S ← {U}
9:                  end if
10:             end if
11:         end for
12:     end for
13: end procedure
```

D. Density-based Clustering

In this section, we present the sub-trajectory clustering algorithm to group them based on the combination of their spatial and non-spatial features.

The aim of density-based clustering algorithm is to identify dense regions that are separated by low-density region. Two most important characteristics is that, they can identify clusters with arbitrary shape and they can find outliers [8].

In this phase we consider two issues,

1) We define two different distance functions. First distance function $dist_s$ aims to find the spatial proximity between two sub-trajectories. The second distance function $dist_{ns}$ is to find their non-spatial attribute similarity.
2) We cluster them considering both distance functions.

1) ***Distance function*** $dist_s$ ***(spatial distance)****:* We define a new distance measure to find the spatial proximity of two sub-trajectories. This distance function is based on the aggregated nearest neighbour distance between the points of the trajectories.

We are given two sub-trajectories S_a and S_b. Here $|S_a|$ denotes the length of S_a and $|S_b|$ denotes the length of S_b. If $|S_a| \geq |S_b|$, the spatial distance between two sub-trajectories S_a and S_b is defined as

$$dist_s(S_a, S_b) = \frac{1}{|S_a|} \sum_{i=1}^{|S_a|} \min_{j=\{1,2,\cdots,|S_j|\}} \{d(S_{ai}, S_{bj})\} \quad (1)$$

That is, for each point in S_a, we find the minimum distance to a point in S_b, and then calculate the average of these distances.

Example: Consider the two sub-trajectory S_a and S_b in Figure 5. Here $|S_a| \geq |S_b|$. Hence,

$dist_s(S_a, S_b) = \frac{1}{6}$ { $d(s_{a1}, s_{b1}) + d(s_{a2}, s_{b1}) + d(s_{a3}, s_{b1}) + d(s_{a4}, s_{b2}) + d(s_{a5}, s_{b3}) + d(s_{a6}, s_{b3})$ }

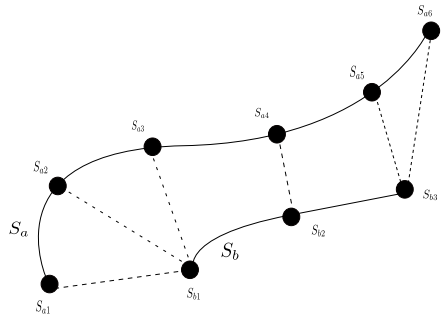

Fig. 5. Spatial distance between two sub-trajectories.

2) ***Distance function*** $dist_{ns}$ ***(non-spatial distance)****:* In Section (C), we describe some non-spatial features. In this paper, we use Euclidean distance method to measure non-spatial attribute similarity among sub-trajectories.

Now we summarize the important notation required for density-based clustering algorithm. This approach is inspired by ST-DBSCAN algorithm [13]. Let D denote the set of all sub-trajectories in the database. This algorithm is based on three threshold : ϵ_s, ϵ_{ns}, $MinS$. The notations we used in the algorithm is:

ϵ_s-Neighbourhood: The ϵ_s-Neighbourhood of a sub-trajectory $N_{\epsilon_s}(S_i)$ is defined by $\{S_j \in D | dist_s(S_i, S_j) \leq \epsilon_s \}$.

ϵ_{ns}-Neighbourhood: The ϵ_{ns}-Neighbourhood of a sub-trajectory $N_{\epsilon_{ns}}(S_i)$ is defined by $\{S_j \in D | dist_{ns}(S_i, S_j) \leq \epsilon_{ns} \}$.

Neighbourhood: The Neighbourhood of a sub-trajectory $N(S_i)$ is defined by $\{S_j \in N_{\epsilon_s}(S_i) \cap N_{\epsilon_{ns}}(S_i) \}$.

Core object: A sub-trajectory S_i is considered as a core object if $|N(S_i)| \geq MinS$.

Border sub-trajectory: A sub-trajectory is considered as a border object if it is not a core object but density reachable from at least one core object.

Directly-density-reachable: A sub-trajectory S_i is directly-density-reachable from another sub-trajectory S_j with respect to ϵ_s, ϵ_{ns} and $MinS$ if 1) $S_i \in N(S_j)$ and 2) $|N(S_j)| \geq MinS$.

Density-reachable: A sub-trajectory S_i is density-reachable from another sub-trajectory S_j with respect to ϵ_s, ϵ_{ns} and $MinS$ if there are a chain of objects S_j, S_{j-1},S_{j-2},$\cdots\cdots$ $S_{i+1}, S_i \in D$ such that S_k is directly density reachable from L_{k+1} with respect to ϵ_s, ϵ_{ns} and $MinS$.

Density-connected: A sub-trajectory S_i is density-connected to a sub-trajectory S_j with respect to ϵ_s, ϵ_{ns} and $MinS$ if there is sub-trajectory $S_k \in D$ such that both S_i and S_j are density reachable from S_k with respect to ϵ_s, ϵ_{ns} and $MinS$.

Density-based cluster: A cluster C of sub-trajectories is a non-empty subset of D satisfying the following condition:

- $\forall S_i$,S_j: if $S_i \in C$ and S_j is density-reachable from S_i, then $S_j \in C$.
- $\forall S_i, S_j \in C$: S_i is density-connected to S_j with respect to ϵ_s, ϵ_{ns} and $MinS$.

3) ***Clustering Algorithm****:* In Algorithm 5, we present the clustering algorithm. Given a set D of sub-trajectories, this algorithm generates a set of clusters based on three threshold values: ϵ_s, ϵ_{ns} and $MinS$. ϵ_s determines object's spatial neighbourhood area and ϵ_{ns} determines it's non-spatial neighbourhood area. Based on these two threshold, this algorithm determines the neighbour sub-trajectories for each sub-trajectory.

Algorithm 5 density-based clustering

Inputs
D : a set of trajectory segments
ϵ_s : Maximum distance to find spatial neighbour
ϵ_{ns} : Maximum distance to find non-spatial neighbour
$MinS$: Minimum number of sub-trajectories necessary to form a cluster
Output:
C : a set of clusters

```
1: procedure DENSITY-CLUSTER(D, ε_s, ε_ns, MinS)
2:     cId ← 1
3:     for i = 1 → n do
4:         if S_i is not visited and not clustered yet then
5:             Mark S_i as visited
6:             X = GET-NEIGHBOUR(S_i)
7:             if |X| < MinS then
8:                 S_i is marked as noise
9:             else
10:                assign clusterId to ∀S ∈ X
11:                Insert all X into the Queue Q
12:                while Q is not Empty do
13:                    pop the current object S_j
14:                    Y = getNeighbour(S_j)
15:                    if |Y| ≥ MinS then
16:                        for ∀s ∈ Y do
17:                            if S_j is not noise then
18:                                if S_j is not in Cluster then
19:                                    assign S_j in CId
20:                                end if
21:                            end if
22:                        end for
23:                    end if
24:                end while
25:                Increment CId by 1
26:            end if
27:        end if
28:    end for
29: end procedure
```

IV. Experiments and Result Evaluation

In this section, we evaluate the effectiveness of our clustering algorithm. We use a real trajectory data set viz., hurricane tracking data [6]. The data comprises Atlantic hurricanes from 1950 to 2000 (50 years). It contains 496 trajectories and 15,998 points. Each track in the data set consists of a sequence of hurricane data sampled at 6-hours intervals. The sample for each instance in a particular trajectory track has latitude,

Algorithm 6 GET-NEIGHBOUR

Inputs
D : a set of trajectory segments
ϵ_s : Maximum distance to find spatial neighbour
ϵ_{ns} : Maximum distance to find non-spatial neighbour
$MinS$: Minimum number of sub-trajectories necessary to form a cluster
Output:
S : a set of sub-trajectories

1: **procedure** GET-NEIGHBOUR($s_i, \epsilon_s, \epsilon_{ns}$)
2: $X \leftarrow$ Spatial Neighbours with respect to ϵ_s and $MinS$
3: $Y \leftarrow$ Non-spatial Neighbours with respect to ϵ_{ns} and $MinS$
4: $N \leftarrow X \cap Y$
5: **return** N
6: **end procedure**

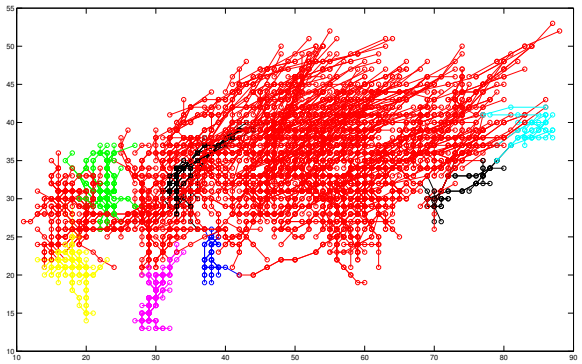

Fig. 6. Clustering results considering spatial distance ϵ_s = 0.002, $MinS$ = 7

longitude, wind speed and wind pressure as its attributes. We did the experiments using MATLAB because of its better visualization.

In our experiments we used latitude and longitude as spatial attributes, and wind speed as non spatial attribute within a particular hurricane trajectory. The first stage of our framework, which is identifying sub-trajectories, resulted in 4044 sub-trajectories.

For these sub-trajectories, the average wind speed for each sub-trajectory is used as its non-spatial attribute. This is mainly motivated by the fact that wind speed is a key characteristic of the storm.

Figs 6, 7 and 8 show the clustering results obtained using only the spatial neighbourhood parameter (viz., ϵ_s), whereas the $MinS$ parameter is 7 throughout the experiments. The parameter ϵ_s determines the size of the spatial neighbourhood to be evaluated for the density of a particular point. From Figs 6 and 7 it can be seen that changing the value of ϵ_s from 0.002 to 0.00195 did not change the clustering result significantly. Whereas, it can be seen from the result in Fig 8 for the value of $\epsilon_s = 0.0018$, that the clustering result changed significantly from those in the Figs 6 and 7 respectively.

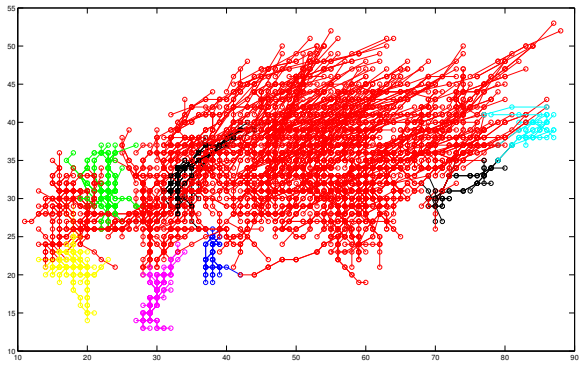

Fig. 7. Clustering results considering only spatial distance with ϵ_s = 0.00195, $MinS$ = 7

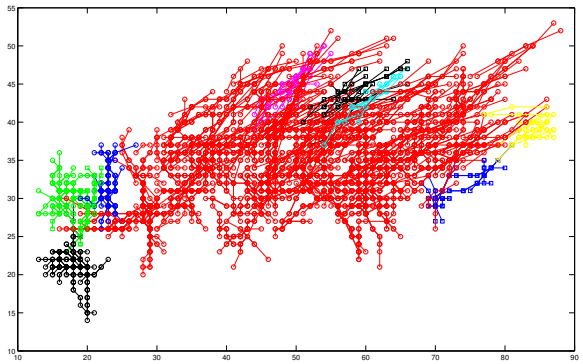

Fig. 8. Clustering results considering only spatial distance with ϵ_s = 0.0018, $MinS$ = 7

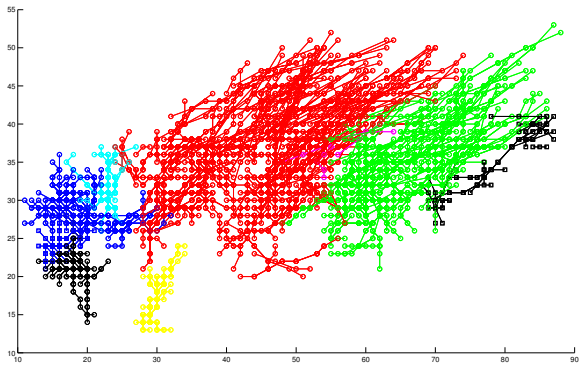

Fig. 9. Clustering results with ϵ_s = 0.002, ϵ_{ns} = 0.2, $MinS$ = 7

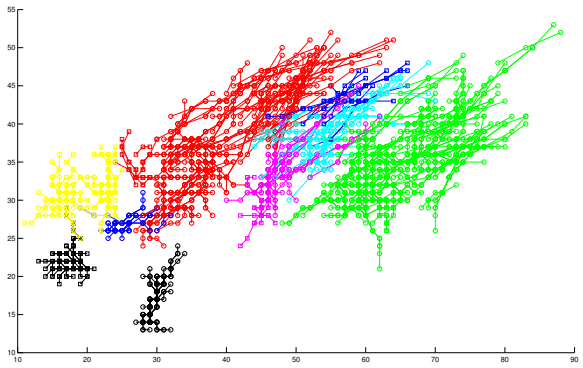

Fig. 10. Clustering results with ϵ_s = 0.00195, ϵ_{ns} = 0.15, $MinS$ = 7

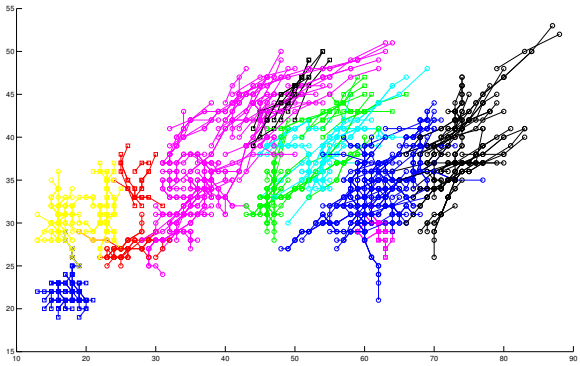

Fig. 11. Clustering results with ϵ_s = 0.0018, ϵ_{ns} = 0.15, $MinS$ = 7

After the experiments with only the spatial neighbourhood parameter, we incorporated the non spatial neighbourhood parameter also. Figs 9, 10, 11 show the results of our clustering algorithm using different parameter values of ϵ_s, ϵ_{ns}. Since we incorporate non spatial attributes also in the clustering process, we used ϵ_{ns} parameter corresponding to the non spatial attribute (viz., wind speed) for the neighbourhood criteria. These parameters are very vital for the performance of the DBSCAN algorithm. For that reason we provide the results for three different sets of these parameters. This parameter specifies the threshold for the dense regions, i.e., a data point will be considered core (dense) only if it has at least $MinS = 7$ data points in its neighbourhood.

Fig 9 uses $\epsilon_s = 0.002$, $\epsilon_{ns} = 0.2$ and $MinS = 7$. For these set of parameters the result shows four major clusters. The biggest cluster of the trajectories is shown in red and is in the central position with respect to the other clusters. This signifies that this particular region witnessed more hurricane activities. Other than this cluster we can see the next significant cluster shown in $green$. The third most significant cluster for this particular parameter set is the one in dark $blue$.

Fig 10 uses $\epsilon_s = 0.00195$, $\epsilon_{ns} = 0.15$ and $MinS = 7$. For these set of parameters the resulting clusters are very different from the ones given in Fig 9. Here the number of clusters is more than the former case and also there is no clear dominating clustering region. The clusters also seem to be overlapping to some extent.

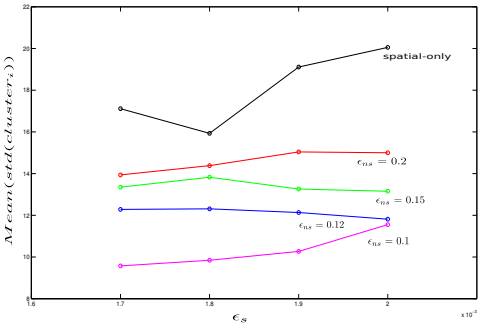

Fig. 12. Clustering result analysis based on non-spatial distance

Finally, Fig 11 uses $\epsilon_s = 0.0018$, $\epsilon_{ns} = 0.15$ and $MinS = 7$. Now the resulting clusters are looking better from the ones given in Fig 10. Here the number of clusters is reduced and the clusters are also more concrete with comparatively less extent of overlapping. Comparing this result with the one in Fig 9, it can be seen that the number of clusters is more in Fig 11. Also, the size of the clusters is comparatively smaller for the current set of parameters.

From the above set of experiments it can be seen that the results of the clustering are very susceptible to the choice of the parameters for the clustering algorithm. The changing behavior of the clustering result from Fig 6 to Fig 11 is natural as with the reduction in the values of ϵ_s, as well as ϵ_{ns}, we are restricting the size of the clusters to include only the more dense regions. In particular, the drastic improvement in the clustering result can be noticed between Fig 8 and Fig 9. This is because of the incorporation of the non spatial neighbourhood parameter which results in more compact clusters.

Fig 12 shows the comparative results of the proposed clustering methodology using different sets of parameters. The clustering using a combination of spatial and non-spatial parameter produces more compact clusters with respect to non-spatial distance.The Y-axis has the $Mean(std(cluster_i))$ for the clustering results, whereas the X-axis shows the different ϵ_s values. We are intuitively using the $Mean(std(cluster_i))$ to compute the compactness of the clusters. This measure computes the average value of the standard deviations of the wind speed, within the respective clusters for a particular set of attributes (viz., ϵ_s and ϵ_{ns}). The highest values (which signifies worst result) for this measure correspond to the case where clustering is done using only the spatial neighbourhood parameter. From Fig 12, we can see the consistent improvement in the results (i.e., reduction in the value of the measure) as spatial neighbourhood parameter is reduced. The best result has been obtained for the $\epsilon_s = 0.0017$ and $\epsilon_{ns} = 0.10$ which is reasonable as this set of parameters forces the clusters to be the most compact compared to the rest.

Fig 13 shows the cluster result analysis based on spatial distance. To measure spatial compactness of each cluster we compute minimum bound rectangle area containing all sub-

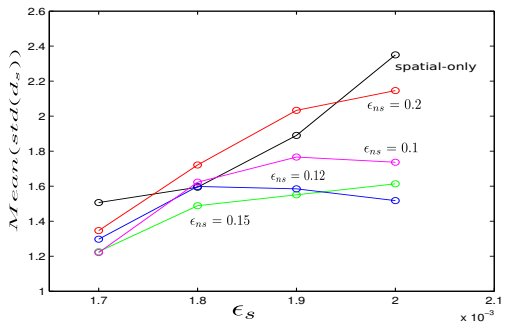

Fig. 13. Clustering result analysis based on spatial distance

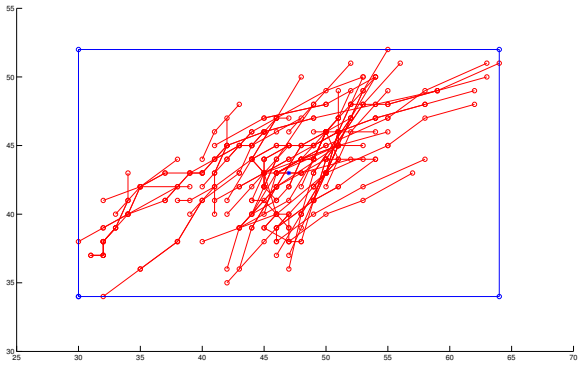

Fig. 14. Cluster region and its centroid point

trajectories of the respective cluster and the centroid point (c) of that area (see Fig 14). Then we compute the average value of standard deviation of spatial distance (d_s) of sub-trajectories from the centroid point (c), where d_s is defined as

$$d_s = \frac{1}{|S_a|} \sum_{i=1}^{|S_a|} \{d(S_{ai}, c)\} \quad (2)$$

where, d_s is the distance of a particular sub-trajectory S_a from the centroid point (c) and $d(S_{ai}, c)$ is the Euclidean distance of a trajectory point S_{ai} from the centroid point. In Fig 13, Y-axis represents the $Mean(std(d_s))$ for each ϵ_s parameter and the X-axis shows the corresponding ϵ_s values.

Fig 13 shows that the spatial compactness of clusters does not suffer when using combination of spatial and non-spatial attributes for clustering. The best spatial compactness is achieved when combination is used.

V. FUTURE WORKS AND CONCLUSION

In this paper, we propose a novel framework to cluster trajectories. In the first phase, we map each trajectory to one-dimensional space. This dimension reduction approach helps to simplify trajectories representation and reduce the cost of comparison with other trajectories. In the second phase, we present our own algorithm to identify all common sub-trajectories based on spatial proximity. In the third phase, we extract significant non-spatial features from them. Finally, we present a nearest neighbour based approach to measure spatial distance of sub-trajectories. Then we cluster all sub-trajectories based on the combination of spatial and non-spatial features.

We perform our experiment on real life hurricane track data. We then compare the clustering that combines the spatial and non spatial attributes, with the clustering based on spatial attributes only. The combined approach always produces better cluster compactness for non-spatial distance, and spatial compactness is not adversely affected.

Spatial datasets are quite large and it is an open research issue to handle big data efficiently. In our future work, we plan to design an algorithm in a distributed framework. We will analyse some other trajectory data sets like vehicle and animal movement. In this approach we do not consider temporal information. We plan to include temporal information during clustering.

REFERENCES

[1] J. Gudmundsson, A. Thom, and J. Vahrenhold, "Of motifs and goals: mining trajectory data," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 129–138.

[2] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 593–604.

[3] J. Hsieh, S.-L. Yu, and Y.-S. Chen, "Trajectory-based video retrieval by string matching," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 4. IEEE, 2004, pp. 2243–2246.

[4] Y. Yanagisawa, J.-i. Akahani, and T. Satoh, "Shape-based similarity query for trajectory of mobile objects," in *Mobile data management*. Springer, 2003, pp. 63–77.

[5] C.-S. Chen, C. F. Eick, and N. J. Rizk, "Mining spatial trajectories using non-parametric density functions," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2011, pp. 496–510.

[6] http://weather.unisys.com/hurrricane/atlantic/index.html.

[7] S. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 63–72.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd, 1996.

[9] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2009, pp. 286–295.

[10] Y. Huang, L. Zhang, and P. Zhang, "A framework for mining sequential patterns from spatio-temporal event data sets," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 4, pp. 433–448, 2008.

[11] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[12] A. M. Cheriyadat and R. J. Radke, "Detecting dominant motions in dense crowds," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 4, pp. 568–581, 2008.

[13] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[14] D. Gusfield, *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, 1997.

[15] S. Shekhar and S. Chawla, *Spatial databases: a tour*. Prentice Hall Englewood Cliffs, 2003, vol. 2003.