TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering *

Jae-Gil Lee, Jiawei Han, Xiaolei Li, Hector Gonzalez
Department of Computer Science
University of Illinois at Urbana-Champaign
{jaegil, hanj, xli10, hagonzal}@uiuc.edu

ABSTRACT

Trajectory classification, *i.e.*, model construction for predicting the class labels of moving objects based on their trajectories and other features, has many important, real-world applications. A number of methods have been reported in the literature, but due to using the *shapes* of *whole* trajectories for classification, they have limited classification capability when discriminative features appear at *parts of trajectories* or are *not relevant to the shapes of trajectories*. These situations are often observed in long trajectories spreading over large geographic areas.

Since an essential task for effective classification is generating discriminative features, a feature generation framework *TraClass* for trajectory data is proposed in this paper, which generates a hierarchy of features by partitioning trajectories and exploring two types of clustering: (1) *region-based* and (2) *trajectory-based*. The former captures the higher-level region-based features without using movement patterns, whereas the latter captures the lower-level trajectory-based features using movement patterns. The proposed framework overcomes the limitations of the previous studies because trajectory partitioning makes discriminative parts of trajectories identifiable, and the two types of clustering collaborate to find features of both *regions* and *sub-trajectories*. Experimental results demonstrate that *TraClass* generates high-quality features and achieves high classification accuracy from real trajectory data.

1. INTRODUCTION

Trajectory data are ubiquitous in the real world. Recent progress on satellite, sensor, RFID, video, and wireless technologies has made it possible to systematically track object movements and collect huge amounts of trajectory data, *e.g.*, animal movement data, vessel positioning data, and hurricane tracking data. Accordingly, there is an ever-increasing interest in performing data analysis over trajectory data [11, 12, 14]. Since classification has played a crucial role in data analysis [9], an effective classification method for trajectory data is needed urgently.

Trajectory classification is defined as the process of predicting the class labels of moving objects based on their trajectories and other features. There are many important, real-world applications driven by real need. We present two application scenarios.

1. *Vessel classification from satellite images [7]*

   Vessel detection and classification from satellite imaging sensors is or could be used for a number of applications: fishery control, pollution control, border control including illegal immigration and smuggling, maritime safety, control of off-shore activities, search and rescue, security of maritime trade routes, and anti-terrorism. According to the report published in 2006, vessel detection in synthetic aperture radar (SAR) images is fairly accurate, but it is beyond today's capability to derive a vessel type from its SAR signature [7]. Most systems use a size estimate for classifying vessels, so small vessels are very hard to classify. Although the trajectories of vessels can be extracted accurately, researchers in this field do not even use them for classification.

2. *Classification of trace gas measurements [2]*

   Trace gas (*e.g.*, ozone) concentration is closely related to the immediate history of the air before arriving at the sampling point. Thus, trace gas concentration can be estimated using wind trajectories. This is actually a classification problem, where the class label is a range of trace gas concentrations. Such classification is useful for sites where air-mass back trajectories are available, but no trace gas concentration has been measured [2]. Meteorologists use a straightforward classification technique based on Euclidean distance between trajectories.

A number of trajectory classification methods have been proposed mainly in the fields of pattern recognition [1], bioengineering [16], and video surveillance [6, 15]. Besides, similar problems exist in the field of time-series classification [20, 21]. A common characteristic of earlier methods is that they use the *shapes* of *whole* trajectories to do classification, *e.g.*, by modeling a whole trajectory with a single mathematical function such as the hidden Markov model (HMM). Although a few methods partition trajectories, the purpose of

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678 and BDI-05-15813, and by the Boeing company. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08

their partitioning is just to approximate or smooth trajectories before using the HMM [1, 6].

Since an essential task for effective classification is generating discriminative features, attention is focused in this paper on feature generation for trajectories. Our framework is motivated by two observations. First, discriminative features are likely to appear at *parts* of trajectories, not at whole trajectories. Second, discriminative features appear not only as common movement patterns, but also as *regions*. These situations are often observed in long trajectories spreading over large geographical areas.

Example 1. Let's consider vessel trajectories that move from port A to port B in Figure 1. Container ships stop at a container port to load cargos, tankers stop at a refinery to load petrochemicals, and fishing boats sail freely in a fishery. Assume that container ships and tankers cannot cross the fishery since water is too shallow.

Typical examples of our observations are shown in Figure 1. (1) Parts of trajectories *near the container port* and *near the refinery* enable us to distinguish between container ships and tankers even if they share common long paths. (2) Those *in the fishery* enable us to recognize fishing boats even if they have no common path there. Notice that such discriminative features are represented by *regions* and *sub-trajectories*. Our framework aims at discovering both of them. On the other hand, earlier methods might not achieve high classification accuracy due to lack of ability to discover them since the overall shapes of whole trajectories are similar to each other. □

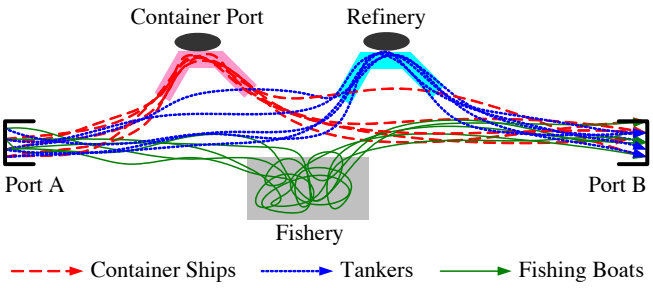


Figure 1: Two types of discriminative features for trajectories.

A feature generation framework *TraClass* for trajectories of free moving objects is proposed in this paper, which performs hierarchical *region-based* and *trajectory-based* clustering after trajectory partitioning. Each feature type in Example 1 is discovered by each clustering. An overview will be provided in Section 2.2.

(1) *Region-based clustering* discovers regions that have trajectories mostly of one class regardless of their movement patterns.

(2) *Trajectory-based clustering* discovers sub-trajectories that indicate common movement patterns of each class.

The collaboration between the two types of clustering leads to discovery of both types of discriminative features, boosting classification accuracy significantly. Since trajectory partitioning precedes clustering, discriminative parts of trajectories become identifiable. Furthermore, *TraClass* is more powerful than earlier methods since it can also discover whole-trajectory features by linking sub-trajectory features as will be explained in Section 5.4.

In summary, the contributions of this paper are as follows:

- We propose a feature generation framework *TraClass* for trajectories, which explores two types of clustering. To the best of our knowledge, this is the most comprehensive framework reported in the literature.
- We propose the notion and algorithm of region-based clustering. The problem of region-based clustering is formalized using the minimum description length (MDL) [8] principle, and an efficient approximate algorithm is developed to find the near-optimal clustering.
- We propose the notion and algorithm of trajectory-based clustering. The procedure of trajectory-based clustering is based on the partition-and-group framework proposed by Lee *et al.* [12]. The partition-and-group framework is extended for classification purposes such that the class labels are incorporated into clustering.
- We demonstrate, by using three real data sets, that classification using the features generated by *TraClass* is very accurate.

The rest of the paper is organized as follows. Section 2 gives an overview of our feature generation framework. Section 3 presents a trajectory partitioning algorithm. Section 4 proposes a region-based clustering algorithm. Section 5 proposes a trajectory-based clustering algorithm. Section 6 explains our classification strategy. Section 7 presents the results of experimental evaluation. Section 8 discusses related work. Finally, Section 9 concludes the study.

2. TRAJECTORY FEATURE GENERATION

2.1 A Working Example

Let us begin by presenting a working example shown in Figures 2 and 3. Suppose there is a set of trajectories from two classes c_1 and c_2, where the trajectories of c_1 are represented by solid lines, and those of c_2 by dashed lines.

1. *Region-based clustering* (Figure 2): First, regions having one major (dominating) class are discovered as in (1). The regions **B**, **F**, and **H** are said to be homogeneous in the sense that they contain trajectories mostly of the same class. Second, the non-homogeneous regions **D** and **E** are recursively quantized to find more of homogeneous regions. The region **J** is found to be homogeneous within **E** as in (2). These homogeneous regions are used as *region-based clusters*. Then, parts of trajectories in non-homogeneous regions are passed to the next step.
2. *Trajectory-based clustering* (Figure 3): Third, common movement patterns of each class are discovered from non-homogeneous regions as in (3). The patterns **3**, **4**, **5**, and **6** are said to be discriminative in the sense that they are different from those of the other class. Fourth, the non-discriminative patterns **1** and **2** are repeatedly investigated in finer granularity to find more of discriminative patterns. The horizontal movements are now represented by two patterns for each class rather than one. The patterns **7**, **8**, **9**, and **10** newly discovered are discriminative as in (4). These discriminative patterns are used as *trajectory-based clusters*.

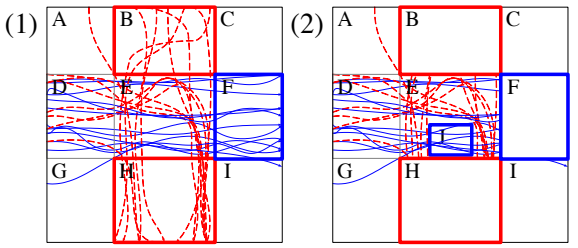


Figure 2: An example of region-based clustering.

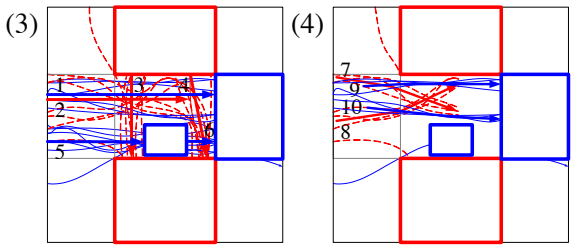


Figure 3: An example of trajectory-based clustering.

2.2 Overall Framework

Figure 4 outlines the feature generation framework *TraClass*. Prior to clustering, each trajectory is partitioned into a set of trajectory partitions. First, *region-based clustering* is performed recursively as long as homogeneous regions of reasonable size are found. The trajectory partitions that are not covered by homogeneous regions are passed to the next step. Second, *trajectory-based clustering* is performed repeatedly as long as discriminative clusters are found. Obviously, this collaborative hierarchical clustering enables us to detect more of high-quality features.

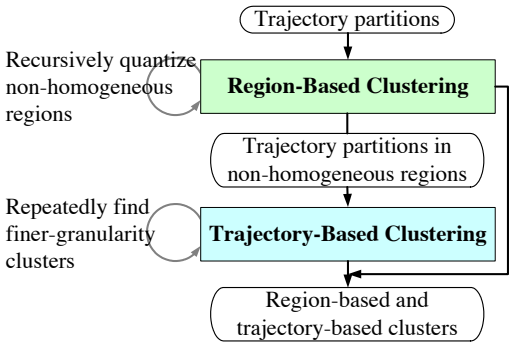


Figure 4: The procedure of hierarchical region-based and trajectory-based clustering.

It is worthwhile to note that exploring the two types of clustering generates a hierarchy of features because region-based clustering discovers higher-level (more general) features than trajectory-based clustering due to not using movement patterns. In addition, within each clustering, there exists a hierarchy from larger clusters to smaller ones. Overall, we generate features in a *top-down* fashion: higher-level features are preferred to lower-level ones since the former are more effective for classification and, at the same time, cheaper to use than the latter [19].

2.3 Problem Statement

We develop a feature generation framework for trajectories of free moving objects. Given a set of *trajectories* $\mathcal{I} = \{TR_1, \ldots, TR_{num_{tra}}\}$, with each trajectory associated with a *class* $c_i \in \mathcal{C} = \{c_1, \ldots, c_{num_{cla}}\}$, our framework generates a set of *features* for trajectory classification, where the trajectory and feature are defined as follows.

A *trajectory* is a sequence of 2-dimensional points and is denoted as $TR_i = p_1 p_2 p_3 \cdots p_j \cdots p_{len_i}$ $(1 \le i \le num_{tra})$. A trajectory $p_{\nu_1} p_{\nu_2} \cdots p_{\nu_k}$ $(1 \le \nu_1 < \nu_2 < \cdots < \nu_k \le len_i)$ is called a *sub-trajectory* of TR_i. A *trajectory partition* is a sub-trajectory $p_i p_j$ $(i < j)$ of length 2. The whole trajectory belongs to the same one class, so does every trajectory partition from the trajectory. Figure 5 shows an example of a trajectory and its trajectory partitions.

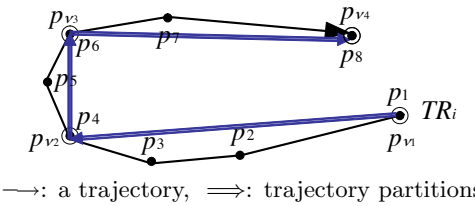


Figure 5: An example of a trajectory and its trajectory partitions.

A *feature* is either a region-based cluster or a trajectory-based cluster. Informally, a *region-based cluster* is a set of trajectory partitions of the same class within a rectangular region regardless of their movement patterns; a *trajectory-based cluster* is a set of trajectory partitions of the same class which share a common movement pattern. Formal definitions will be given in Sections 4.1 and 5.1, respectively. These features, generated by two types of clustering, are provided to a classifier.

3. TRAJECTORY PARTITIONING

3.1 Preliminaries

Trajectory partitioning as well as trajectory-based clustering are based on the partition-and-group framework [12], which consists of the following two phases:

(1) The *partitioning* phase: Each trajectory is partitioned into a set of line segments (*i.e.*, trajectory partitions) whenever its moving direction changes rapidly. The problem of finding the optimal set of such partitioning points is formulated by the MDL principle. An $O(n)$ approximate algorithm is proposed to efficiently find the near-optimal partitioning. See Appendix A for details.

(2) The *grouping* phase: Similar line segments are grouped into a cluster using a density-based clustering method analogous to DBSCAN [5]. A distance function for line segments $dist(L_i, L_j)$ is designed to define the density. The basic notions of density-based clustering for points are changed to those for line segments as follows. Two parameters ε and $MinLns$ are introduced.

- The *ε-neighborhood* of a line segment L_i is $N_\varepsilon(L_i) = \{L_j \in \mathcal{L} \mid dist(L_i, L_j) \le \varepsilon\}$, where $\mathcal{L}$ is the set of trajectory partitions.
- L_i is a *core line segment* if $|N_\varepsilon(L_i)| \ge MinLns$.
- L_i is *directly density-reachable* from L_j if $L_i \in N_\varepsilon(L_j)$ and L_j is a core line segment.

- *Density-reachability* is the transitive closure of directly density-reachability.
- L_i is *density-connected* to L_j if both L_i and L_j are density-reachable from a third line segment. A density-connected set is considered as a *cluster*.

At the final stage of the grouping phase, a model called a *representative trajectory*, which is a sequence of points just like an ordinary trajectory, is generated for each cluster. It is an imaginary trajectory that indicates the major movement pattern of the trajectory partitions belonging to the cluster and is obtained by calculating the average coordinates of those trajectory partitions. The overall procedure explained above is summarized in Figure 6.

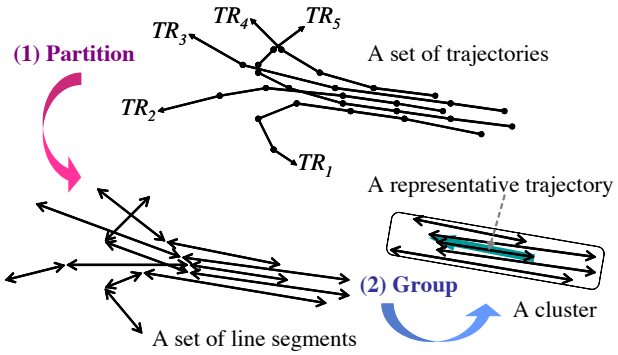
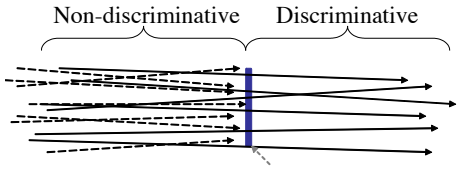


Figure 6: An overall procedure of trajectory clustering in the partition-and-group framework.

The original partition-and-group framework is *extended for classification purposes* so that the class labels are incorporated into partitioning and clustering. Notice that the partition-and-group framework is adapted just as a *component* of *TraClass*. The extension for trajectory partitioning is explained in Section 3.2, and that for trajectory-based clustering in Section 5.2.

3.2 Class-Conscious Partitioning

After each trajectory is partitioned by its movement pattern, some trajectory partitions need to be further partitioned by the class labels. The real interest here is to guarantee that trajectory partitions do not span the class boundaries as illustrated in Figure 7.



→: trajectories of c_1, ⇢: trajectories of c_2

Figure 7: Trajectory partitions that require further partitioning.

Example 2. Figure 7 shows a bunch of trajectory partitions from two classes c_1 and c_2. The trajectory partitions of c_1 overlap those of c_2. As a result, the non-overlapping part is discriminative, whereas the overlapping part is not. Obviously, it is preferable to separate the discriminative part from the non-discriminative part. By virtue of this class-conscious partitioning, the discriminative part can be identified as a cluster. □

The basic idea of *class-conscious partitioning* is to further partition a trajectory partition if the segments between two endpoints have very different class distributions. It ensures that the class distribution is kept uniform along a trajectory partition. To achieve this class-conscious partitioning, a *heterogeneous* trajectory partition is defined through Definitions 1 and 2.

Definition 1. The *class affinity* of a point p is a vector $\mathcal{CA}(p) = \langle f_1, f_2, \ldots, f_{num_{cla}} \rangle$, where f_i is the frequency of trajectory partitions whose class label is equal to i and whose at least one endpoint is within Euclidean distance σ from p. Here, σ is the standard deviation of pairwise distances between all partitioning points.

Basically, if the most prevalent class around one endpoint is different from that around the other endpoint, the trajectory partition is determined to be heterogeneous and is partitioned at between two endpoints.

Definition 2. A trajectory partition $L = p_s p_e$ is *heterogeneous* if $\text{argmax}_k[\mathcal{CA}(p_s)]_k \neq \text{argmax}_k[\mathcal{CA}(p_e)]_k$. Here, $[\mathcal{CA}(\cdot)]_k$ denotes the frequency of trajectory partitions whose class label is equal to k, and $\text{argmax}_k[\mathcal{CA}(\cdot)]_k$ means the class label with the maximum frequency in $\mathcal{CA}(\cdot)$.

Figure 8 shows the algorithm *Class-Conscious Trajectory Partitioning*. The algorithm first partitions each trajectory by its movement pattern as done by Lee *et al.* [12] (lines 1∼3) and then finds heterogeneous trajectory partitions by Definition 2 (lines 4∼6). The heterogeneous ones are examined to find proper partitioning points (lines 7∼11). Recall that a trajectory partition encloses multiple points of a raw trajectory as in Figure 5. Thus, the concept of heterogeneity is applied again to each atomic line segment within the heterogeneous trajectory partition. If a line segment $p_k p_{k+1}$ is heterogeneous, its enclosing trajectory partition is divided into two parts before and after p_{k+1} (lines 8∼10).

4. REGION-BASED CLUSTERING

4.1 Definition of Region-Based Clusters

A *region-based cluster* is formally defined through Definitions 3 and 4. This definition is very intuitive: a region-based cluster contains *many* of trajectory partitions of one *major* class, but very *few* (hopefully, *none*) of trajectory partitions of other *minor* classes. Examples are depicted as thick rectangles in Figure 2.

Definition 3. A region in a 2-dimensional space is *homogeneous* if only one class c_{major} has trajectory partitions from $\geq \psi$ trajectories within the region, but all other classes do not. The class c_{major} is called the *major* class of the region, and other classes are called *minor* classes.

In the definition above, ψ designates the *minimum population* of the major class in a homogeneous region. ψ typically shares the parameter value with $MinLns$ to reduce the number of parameters to optimize. The two parameters, in fact, play the same role in region-based and trajectory-based clustering.

Algorithm **Class-Conscious Trajectory Partitioning**

INPUT: A set $\mathcal{I}$ of trajectories
OUTPUT: A set $\mathcal{L}$ of trajectory partitions
ALGORITHM:
01: **for each** $TR \in \mathcal{I}$ **do**
02: Partition TR by its movement pattern;
03: Accumulate trajectory partitions into a set $\mathcal{L}$;
/* Find heterogeneous trajectory partitions */
04: **for each** $L \in \mathcal{L}$ **do**
05: **if** L is *heterogeneous* by Definition 2 **then**
06: Add L into a set $\mathcal{H}$;
/* Further partition the heterogeneous ones */
07: **for each** $L \in \mathcal{H}$ **do**
/* Suppose L encloses $p_{\nu_i} p_{\nu_i+1} p_{\nu_i+2} \cdots p_{\nu_{i+1}}$ */
08: **for each** $k \in [\nu_i, \nu_{i+1} - 1]$ **do**
09: **if** $p_k p_{k+1}$ is *heterogeneous* by Definition 2 **then**
10: Partition L at the point p_{k+1};
11: Replace L with new trajectory partitions;

Figure 8: A class-conscious algorithm for partitioning trajectories.

Definition 4. A *region-based cluster* is a set of trajectory partitions of the major class within a homogeneous rectangular region.

A rectangular region that encloses trajectory partitions is used to represent the region-based cluster. For ease of computation, we regard that a region encloses a trajectory partition if its center point is located inside the region.

4.2 Construction of the Grid Structure

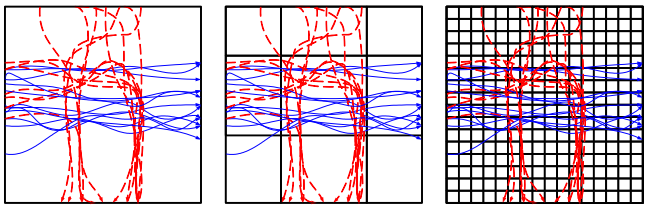
To find homogeneous regions as much as possible, *TraClass* uses a multi-resolution grid structure, which quantizes the domain space into a finite number of cells. The $\mathbb{X}$ and $\mathbb{Y}$ axes are partitioned separately, and then, cells are generated by crossing the partitions of the $\mathbb{X}$ and $\mathbb{Y}$ axes. After quantization, each region (*i.e.*, cell) is examined to determine if it is homogeneous.

A good quantization of the domain space should possess two desirable properties: *homogeneity* and *conciseness*. Homogeneity means that the class distribution in each region should be as homogeneous as possible; it is required for deriving more region-based clusters. Conciseness means that the number of regions should be as small as possible; it is required for generating larger region-based clusters.

Homogeneity and conciseness are rivalry measures. If the entire domain space is quantized into one region as in Figure 9 (a), homogeneity may become lowest, but conciseness becomes highest. In contrast, if the domain space is quantized into many small regions so that they enclose at most one trajectory partition as in Figure 9 (c), homogeneity becomes highest, but conciseness becomes lowest. Thus, it is necessary to find a good tradeoff between the two properties as in Figure 9 (b).

4.2.1 Formalization Using the MDL Principle

The MDL cost consists of two components [8]: $L(H)$ and $L(D|H)$. Here, H means the hypothesis, and D the data. The two components are informally stated as follows [8]: "$L(H)$ is the length, in bits, of the description of the hypothesis; and $L(D|H)$ is the length, in bits, of the description of the data when encoded with the help of the hypothesis." The best hypothesis H to explain D is the one which minimizes the sum of $L(H)$ and $L(D|H)$.



(a) Lowest/Highest. (b) High/High. (c) Highest/Lowest. (Homogeneity/Conciseness)

Figure 9: Comparison of homogeneity and conciseness between two extreme cases.

In our quantization problem, H corresponds to the partitions of the $\mathbb{X}$ and $\mathbb{Y}$ axes, and D to the class labels of trajectory partitions. As a result, finding a good quantization translates to finding the best hypothesis using the MDL principle. Before proceeding, the notation for region-based clustering is summarized in Table 1.

Table 1: The notation for region-based clustering.

SYMBOL	DEFINITION
$num_{\mathbb{X}}$	the number of partitions of the $\mathbb{X}$ axis
$num_{\mathbb{Y}}$	the number of partitions of the $\mathbb{Y}$ axis
$size_{\mathbb{X},i}$	the size of the i-th partition of the $\mathbb{X}$ axis
$size_{\mathbb{Y},j}$	the size of the j-th partition of the $\mathbb{Y}$ axis
$R_{i,j}$	the region obtained by crossing the i-th $\mathbb{X}$-axis partition and the j-th $\mathbb{Y}$-axis partition
$N(R_{i,j})$	the total number of trajectory partitions located inside $R_{i,j}$
$N_k(R_{i,j})$	the number of trajectory partitions with the class label k inside $R_{i,j}$
$C(R_{i,j})$	the code cost of $R_{i,j}$

1. $L(H)$ is formulated by Eq. (1). The first and second terms are required to describe the number of partitions of the $\mathbb{X}$ and $\mathbb{Y}$ axes, respectively. Here, $\log^*$ is the universal code length for integers.[1] The third and fourth terms are required to describe the size of each partition of the $\mathbb{X}$ and $\mathbb{Y}$ axes, respectively.
2. $L(D|H)$ is formulated by Eq. (2). The first term encodes the number of trajectory partitions within $R_{i,j}$. The second term $C(R_{i,j})$, called the *code cost*, represents the number of bits required to transmit the class labels of trajectory partitions within $R_{i,j}$.[2] If only one class exists in $R_{i,j}$ (*i.e.*, $R_{i,j}$ is purely homogeneous), $C(R_{i,j})$ becomes zero, but if many classes are mixed up in $R_{i,j}$, $C(R_{i,j})$ becomes high. Finally, $L(D|H)$ is obtained by summing up the two values of every $R_{i,j}$.

[1] It can be shown that $\log^*(n) = \log_2 c + \log_2 n + \log_2 \log_2 n + \cdots$, where the sum only includes positive terms and $c \approx 2.865$ [13]. If n is large, $\log^*(n) \approx \log_2 n$.
[2] Recall information entropy theory. Suppose X can have one of m values $\{v_1, v_2, \ldots, v_m\}$ with probability $P(X = v_i) = p_i$. Then, the smallest number of bits, on average per symbol, needed to transmit a stream of symbols drawn from X's distribution is $H(X) = -\sum_{j=1}^{m} p_j \log_2 p_j$ [17].

$$L(H) = \log^*(num_{\mathbb{X}}) + \log^*(num_{\mathbb{Y}}) + \sum_{i=1}^{num_{\mathbb{X}}} \lceil \log_2 size_{\mathbb{X},i} \rceil + \sum_{j=1}^{num_{\mathbb{Y}}} \lceil \log_2 size_{\mathbb{Y},j} \rceil \quad (1)$$

$$L(D|H) = \sum_{i=1}^{num_{\mathbb{X}}} \sum_{j=1}^{num_{\mathbb{Y}}} \{\lceil \log_2(N(R_{i,j})+1) \rceil + C(R_{i,j})\}, \quad (2)$$

$$\text{where } C(R_{i,j}) = -N(R_{i,j}) \sum_{k=1}^{num_{cla}} \frac{N_k(R_{i,j})}{N(R_{i,j})} \log_2 \frac{N_k(R_{i,j})}{N(R_{i,j})}$$

Finding the quantization that minimizes $L(H) + L(D|H)$ is exactly the tradeoff between homogeneity and conciseness in the sense that $L(H)$ measures the degree of conciseness, and $L(D|H)$ that of homogeneity. Although theoretically pleasing, our problem is computationally expensive: since even the optimal $num_{\mathbb{X}}$ and $num_{\mathbb{Y}}$ are unknown, it is required to find the best quantization for every possible pair of them. Thus, an approximate algorithm is developed in the next section.

4.2.2 An Approximate Algorithm

Figure 10 shows the algorithm *Region-Based Clustering*. At the beginning, one big partition enclosing the entire range is assumed to exist for each axis. The algorithm progressively finds a better partitioning alternately for the $\mathbb{X}$ axis and for the $\mathbb{Y}$ axis as long as the MDL cost decreases. Among the current set of partitions of the $\mathbb{X}$ axis, the algorithm selects the one that has the maximum code cost (line 4) and divides it into two parts in order to decrease the MDL cost as much as possible (line 5). Consequently, a new partitioning point is determined so as to minimize the sum of the code costs of two new partitions. Then, if this new partitioning makes the MDL cost decrease, the set of partitions and the minimum MDL cost are updated accordingly (lines 6∼9). The same procedure is applied to the $\mathbb{Y}$ axis. These procedures are repeated until there is no decrease in both $\mathbb{X}$ and $\mathbb{Y}$ axes (lines 11∼12). Finally, all homogeneous regions are retrieved according to Definition 3.

LEMMA 1. *The time complexity of the algorithm in Figure 10 is* $O((num_{\mathbb{X}} + num_{\mathbb{Y}}) \cdot n)$*, where* n *is the total number of trajectory partitions in a database.*

PROOF: The algorithm exits the loop after executing lines 4∼9 by $(num_{\mathbb{X}} + num_{\mathbb{Y}})$ times because $num_{\mathbb{X}}$ (or $num_{\mathbb{Y}}$) is always increased by one. At every execution, the algorithm scans the set of trajectory partitions to calculate the code cost of each region $R_{i,j}$ under the current quantization. This calculation requires accessing n trajectory partitions. □

Example 3. Figure 11 shows a step-by-step procedure of generating region-based clusters from the set of trajectories in Figure 2. In each figure, the shaded rectangle indicates the worst partition that has the maximum code cost, and the arrow a new partitioning point of the worst partition. These selection and partition are performed alternately for the $\mathbb{X}$ axis and for the $\mathbb{Y}$ axis. After the fourth partitioning, the current quantization is determined to be optimal because further partitioning increases the MDL cost. □

4.3 Recursive Quantization and Merging

TraClass recursively quantizes the regions that are not homogeneous. This recursive quantization enables us to find more (small-sized) region-based clusters. In more detail, the regions where two or more classes have trajectory partitions from $\geq \psi$ trajectories (*e.g.*, **D** and **E** in Figure 2) are qualified for recursive quantization. If regions contain very few or none of trajectories (*e.g.*, **A**, **C**, **G**, and **I** in Figure 2), such regions are not investigated anymore since they do not provide meaningful information for classification.

Recursive quantization over non-homogeneous regions can be done by the same algorithm, except that initial $\mathcal{X}$ and $\mathcal{Y}$ are generated using the range of the target region. This

---

Algorithm **Region-Based Clustering**

INPUT: A set $\mathcal{L}$ of trajectory partitions
OUTPUT: A set $\mathcal{R}$ of region-based clusters
ALGORITHM:
/* $\mathcal{X}$ (or $\mathcal{Y}$) is a set of partitions of the $\mathbb{X}$ (or $\mathbb{Y}$) axis */
01: $\mathcal{X} := \{[min_{\mathbb{X}}, max_{\mathbb{X}}]\}$, $\mathcal{Y} := \{[min_{\mathbb{Y}}, max_{\mathbb{Y}}]\}$;
02: $minCost := MDL(\mathcal{X}, \mathcal{Y})$; /* $L(H) + L(D|H)$ */
03: **while** *true* **do**
/* $[x^m_{start}, x^m_{end}]$ denotes the m-th partition */
04: Choose the m-th partition from $\mathcal{X}$, where
$$m := \underset{1 \leq i \leq num_{\mathbb{X}}}{\text{argmax}} \sum_{j=1}^{num_{\mathbb{Y}}} C(R_{i,j});$$
05: Find a new partitioning point x^m_{new}, where
$$x^m_{new} := \underset{x^m_{start} < p < x^m_{end}}{\text{argmin}} \sum_{j=1}^{num_{\mathbb{Y}}} \{C(R_{[x^m_{start}, p], j}) + C(R_{[p, x^m_{end}], j})\};$$
/* Divide the worst partition into two */
06: $\mathcal{X}' := \mathcal{X} - \{[x^m_{start}, x^m_{end}]\} \cup \{[x^m_{start}, x^m_{new}], [x^m_{new}, x^m_{end}]\}$;
07: $newCost_{\mathbb{X}} := MDL(\mathcal{X}', \mathcal{Y})$;
/* Check if the MDL cost decreases */
08: **if** $newCost_{\mathbb{X}} < minCost$ **then**
09: $\mathcal{X} := \mathcal{X}'$, $minCost := newCost_{\mathbb{X}}$;
10: Repeat (4)∼(9) for the $\mathbb{Y}$ axis;
11: **if** there is no decrease in both $\mathbb{X}$ and $\mathbb{Y}$ axes **then**
12: Stop repetition; /* *break* */
13: Construct $R_{i,j}$'s ($1 \leq i \leq num_{\mathbb{X}}$, $1 \leq j \leq num_{\mathbb{Y}}$);
14: Output *homogeneous* $R_{i,j}$'s by Definition 3;

Figure 10: An approximate algorithm for region-based clustering.

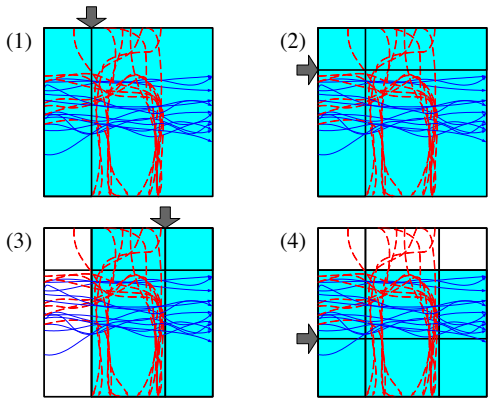


Figure 11: A step-by-step procedure of generating region-based clusters.

algorithm is recursively executed over non-homogeneous regions as long as a homogeneous region has been found after recursive quantization.

After all possible homogeneous regions are discovered, adjacent regions can be merged to form larger regions. Apparently, larger regions are more intuitive to humans and more effective for classification than smaller regions. Suppose there are two homogeneous regions R_{Hi} and R_{Hj}. The two regions are merged if (i) R_{Hi} and R_{Hj} are adjacent to each other; (ii) R_{Hi} and R_{Hj} share the same major class; and (iii) the merger of R_{Hi} and R_{Hj} still forms a rectangle. The first and second conditions are very natural, and the third condition is necessary for easy maintenance.

5. TRAJECTORY-BASED CLUSTERING

5.1 Definition of Trajectory-Based Clusters

A *trajectory-based cluster* is formally defined in Definition 5. This definition is essentially the same as the original definition of a trajectory cluster *except the class constraint*. Examples are depicted as thick arrow lines in Figure 3.

Definition 5. A *trajectory-based cluster* is a density-connected set of trajectory partitions of the same class.

A representative trajectory is generated to represent the trajectory-based cluster. This representative trajectory is also used in defining the distance between two trajectory-based clusters.

5.2 Class-Conscious Grouping

By definition, a trajectory-based cluster should grow using trajectory partitions of the same class. The real interest here is to guarantee that a cluster is derived from only one class. It is obvious that common sub-trajectories composed of different classes are of no use for classification.

5.2.1 Distance Function for Line Segments

The distance function that we use is composed of three components [11, 12]: (i) the *perpendicular distance* ($d_\perp$), (ii) the *parallel distance* ($d_\parallel$), and (iii) the *angle distance* (d_θ). They are intuitively illustrated in Figure 12.

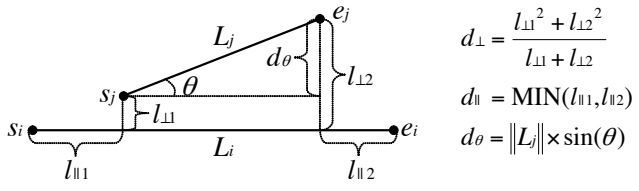


Figure 12: Three components of the distance function for line segments.

The distance function is defined as the weighted sum of the three components: $dist(L_i, L_j) = w_\perp \cdot d_\perp(L_i, L_j) + w_\parallel \cdot d_\parallel(L_i, L_j) + w_\theta \cdot d_\theta(L_i, L_j)$. The weights $w_\perp$, $w_\parallel$, and w_θ are determined depending on applications. We set these weights equally to 1 in default. This default value generally works well in many applications.

5.2.2 Homogeneity of an ε-Neighborhood

To achieve this class-conscious grouping, a *homogeneous* ε-neighborhood is defined through Definitions 6 and 7.

Definition 6. The *class distribution* of an ε-neighborhood $N_\varepsilon(L)$ is a vector $\mathcal{CD}(N_\varepsilon(L)) = \langle f_1, f_2, \ldots, f_{num_{cla}} \rangle$, where f_i is the frequency of trajectory partitions whose class label is equal to i within $N_\varepsilon(L)$.

Definition 7. An ε-neighborhood $N_\varepsilon(L)$ is *homogeneous* if the condition below is satisfied. The class whose label is equal to $\text{argmax}_k[\mathcal{CD}(N_\varepsilon(L))]_k$ is called the *major class* of $N_\varepsilon(L)$, and other classes are called *minor* classes.

(i) the class label of $L = \text{argmax}_k[\mathcal{CD}(N_\varepsilon(L))]_k$ $\wedge$
(ii) $\max[\mathcal{CD}(N_\varepsilon(L))]_k \geq MinLns$ $\wedge$
(iii) $\forall l \neq \text{argmax}_k[\mathcal{CD}(N_\varepsilon(L))]_k, [\mathcal{CD}(N_\varepsilon(L))]_l < MinLns$

After an ε-neighborhood is computed, it is checked to determine if it is homogeneous by Definition 7. If the ε-neighborhood is homogeneous, it is used for trajectory-based clustering. Otherwise, it is immediately discarded.

Example 4. Reconsider the bunch of trajectory partitions in Figure 7. Owing to class-conscious partitioning, the trajectory partitions of c_1 (solid line segments) have been further partitioned as in Figure 13. Let $MinLns$ be 5. We can easily see that $N_\varepsilon(L_1)$ is non-homogeneous since trajectory partitions of two classes are mixed up. In contrast, $N_\varepsilon(L_2)$ is clearly homogeneous. Thus, only the trajectory partitions in $N_\varepsilon(L_2)$ are used for trajectory-based clustering. □

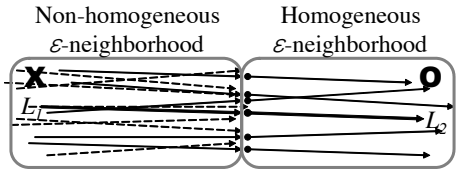


Figure 13: Homogeneity of an ε-neighborhood.

Figure 14 shows the algorithm *Class-Conscious Trajectory Grouping*. The algorithm selects an unmarked trajectory partition L (line 2) and checks if its ε-neighborhood $N_\varepsilon(L)$ is homogeneous by Definition 7 (lines 3∼4). A set of parameter values is maintained for each class, and the appropriate set is dynamically chosen by the class label of L. If $N_\varepsilon(L)$ is homogeneous, the algorithm computes a density-connected set from the trajectory partitions of the major class in $N_\varepsilon(L)$. This density-connected set C constitutes a cluster (lines 5∼8). Otherwise, L is marked as a noise (lines 9∼10). This process is repeated until all trajectory partitions are marked as either a cluster or a noise.

LEMMA 2. *The time complexity of the algorithm in Figure 14 is $O(n^2)$ if a spatial index is not used, where n is the total number of trajectory partitions in a database.*

PROOF: The algorithm has the same time complexity as DBSCAN [5] because both algorithms compute a density-connected set to derive a cluster. □

5.2.3 Analysis of Discriminative Power

After trajectory-based clusters are found, *TraClass* selects those whose discriminative power is high enough for use in classification. However, existing measures of discriminative power, such as the information gain [9], are not adequate since they are originally designed for discrete data on the assumption that the degree of difference between data items

Algorithm **Class-Conscious Trajectory Grouping**

INPUT: (1) A set $\mathcal{L}$ of trajectory partitions,
(2) Two parameters ε and $MinLns$ for each class
OUTPUT: A set $\mathcal{T}$ of trajectory-based clusters
ALGORITHM:
01: Initially unmark $\forall L \in \mathcal{L}$;
02: **while** unmarked $L \in \mathcal{L}$ exists **do**
/* Choose ε and $MinLns$ for the class of L */
03: Compute $N_\varepsilon(L)$;
/* Examine if $N_\varepsilon(L)$ is homogeneous */
04: **if** $N_\varepsilon(L)$ is *homogeneous* by Definition 7 **then**
05: Remove $\forall X \in N_\varepsilon(L)$ of minor classes;
/* Construct a cluster growing from $N_\varepsilon(L)$ */
06: Compute a *density-connected set* C from $N_\varepsilon(L)$;
07: Mark $\forall M \in C$ as a *cluster*;
08: Add C into a set $\mathcal{T}$;
09: **else**
10: Mark L as a *noise*;

Figure 14: A class-conscious algorithm for grouping trajectory partitions.

is not important. Obviously, this assumption is not true in our task: *far-away* located clusters of different classes are more discriminative than *closely* located clusters. Thus, a new measure is necessary for our task.

The new measure relies on the distance among clusters, which is defined by the formal concept of the *Hausdorff distance* [4] that measures how far two compact non-empty subsets of a metric space are from each other. A cluster is represented by its representative trajectory, *i.e.*, a sequence of line segments. Thus, the distance between two clusters is calculated using two sets of line segments. The perpendicular and angle distances for clusters are formulated by Eqs. (3) and (4), respectively.[3] Here, $\|L_i\|$ denotes the length of a line segment, and $\|C_i\|$ the sum of the length of every line segment in C_i's representative trajectory. The weighted sum is used as follows: $dist(C_i, C_j) = w_\perp \cdot d_\perp(C_i, C_j) + w_\theta \cdot d_\theta(C_i, C_j)$.

$$d_\perp(C_i, C_j) = \sum_{L_i \in C_i} \frac{\|L_i\|}{\|C_i\|} \left\{ \sum_{L_j \in C_j} \frac{\|L_j\|}{\|C_j\|} \cdot d_\perp(L_i, L_j) \right\} \quad (3)$$

$$d_\theta(C_i, C_j) = \sum_{L_i \in C_i} \sum_{L_j \in C_j} d_\theta(L_i, L_j) \quad (4)$$

The discriminative power of a cluster is called the *separation gain* and defined in Definition 8. Basically, it is the average distance from a specific cluster to other clusters of different classes. The larger the separation gain of a cluster is, the more likely the cluster is discriminative.

Definition 8. The *separation gain* of a cluster C_i is defined as Eq. (5). Here, $\mathcal{T}$ denotes the set of all clusters, and $CL(\cdot)$ the class label of a cluster.

$$sep\_gain(C_i) = \frac{1}{|\Upsilon(C_i)|} \sum_{C_j \in \Upsilon(C_i)} dist(C_i, C_j), \quad (5)$$
$$\text{where } \Upsilon(C_i) = \{C_j \mid C_j \in \mathcal{T} \wedge CL(C_i) \neq CL(C_j)\}$$

[3]We do not consider the parallel distance here to ignore the positional difference of line segments within a cluster.

Our criterion for selecting discriminative clusters is provided in Lemma 3. A cluster is regarded as *discriminative* if it is well separated from those of different classes by more than the overall separability ξ, which is the *median* of pairwise distances between all clusters.

LEMMA 3. *If $sep\_gain(C_i) > \xi$, at least one cluster of different classes is far from C_i by more than ξ.*

PROOF: We use proof by contradiction. Assume that the distances from C_i to clusters of different classes are all less than or equal to ξ. Then, $sep\_gain(C_i) \leq \xi$ by Definition 8, contradicting our assumption. □

Notice that the median is chosen instead of the average to quantify the overall separability. The reason is that the average is found to be less robust to outliers than the median since some extremely large distances could make the threshold too high.

5.3 Repeated Execution and Parameter Value Selection

Once a trajectory-based cluster is selected by Lemma 3, the trajectory partitions in the cluster are not considered in subsequent clustering since they are already covered by a discriminative cluster. Thus, *TraClass* repeatedly performs clustering over only *uncovered* trajectory partitions in order to generate a larger number of smaller clusters by adjusting parameter values. This repetitive clustering enables us to find more (probably discriminative) clusters. Since it is required to have a reasonable interval of parameter values, we provide a heuristic on how to determine initial parameter values and how to adjust them.

The heuristic is based on the following observations. In the worst clustering, $|N_\varepsilon(L)|$ tends to be uniform. More specifically, for too small an ε, $|N_\varepsilon(L)|$ becomes 1 for almost all trajectory partitions; for too large an ε, it becomes num_{tp} for almost all trajectory partitions, where num_{tp} is the total number of trajectory partitions. Thus, the entropy becomes maximum. In contrast, in a good clustering, $|N_\varepsilon(L)|$ tends to be skewed. Thus, the entropy becomes smaller.

The entropy formula is defined as Eq. (6). The value of ε that minimizes $H(X)$, ε_{min}, is efficiently obtained by a simulated annealing [10] technique. Then, the heuristic constructs an interval $[\varepsilon_{vbegin}, \varepsilon_{vend}]$ centered at ε_{min}, where the difference of the entropy from the minimum is very small ($< 1\%$). This interval can be easily derived because the entropy plot usually forms a valley at ε_{min}.

$$H(X) = \sum_{i=1}^{n} p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \quad (6)$$
$$\text{where } p(x_i) = \frac{|N_\varepsilon(x_i)|}{\sum_{j=1}^{n} |N_\varepsilon(x_j)|} \text{ and } n = num_{tp}$$

The initial value for the parameter ε is set to be ε_{vend}. In addition, the initial value for the parameter $MinLns$ is set to be $(avg_{|N_\varepsilon(L)|} + 1 \sim 3)$, where $avg_{|N_\varepsilon(L)|}$ is the average of $|N_\varepsilon(L)|$ at $\varepsilon = \varepsilon_{vend}$. This is natural since $MinLns$ should be greater than $avg_{|N_\varepsilon(L)|}$ to discover meaningful clusters. Computing $avg_{|N_\varepsilon(L)|}$ induces no additional cost since it can be done while computing $H(X)$.

During the subsequent iterations, the value of ε is decreased with the value of $MinLns$ fixed. As the value of ε is decreased, a larger number of smaller clusters are discovered. The value at the i-th iteration ε^i $(i \geq 1)$ is determined

by Eq. (7), where num_{max_iter} is the maximum number of iterations to be performed. In all tests we conducted, the number of clusters is shown to decrease rapidly after the second iteration. Thus, it is expected that $num_{max\_iter} = 3 \sim 4$ is a reasonable setting for many real situations.

$$\varepsilon^i = \varepsilon_{vend} - \frac{(\varepsilon_{vend} - \varepsilon_{vbegin})}{num_{max\_iter}} \times (i-1) \quad (7)$$

Notice that parameter values are derived separately for each class. In the process of class-conscious grouping, the appropriate set of parameter values is dynamically chosen based on the class label of the current trajectory partition.

5.4 Cluster Link Generation

In addition to deriving highly discriminative trajectory-based clusters, we develop a technique to further improve classification accuracy. Due to the *sequential nature* of trajectory data, a sequence of clusters is likely to hold important information for use in classification.

To take advantage of the sequential nature, combined features are generated from a set of clusters. A *combined feature* is defined as a sequence of connectable clusters or connectable combined-features. Definition 9 presents the condition that needs to be satisfied to be *connectable*.

Definition 9. Two clusters C_i and C_j are *connectable* if Ineq. (8) is satisfied. Here, $TR(C)$ denotes the set of trajectories from which trajectory partitions in the cluster C have been extracted.

$$\frac{|TR(C_i) \cap TR(C_j)|}{\text{MIN}(|TR(C_i)|, |TR(C_j)|)} \geq \chi \quad (8)$$

Let F_i be $\langle C_{F_{i,1}}, \ldots, C_{F_{i,k}} \rangle$ and F_j be $\langle C_{F_{j,1}}, \ldots, C_{F_{j,k}} \rangle$. Then, two combined features F_i and F_j are *connectable* if $\forall l \in [1, k-1], C_{F_{i,l+1}} = C_{F_{j,l}}$ and Ineq. (9) is satisfied.

$$\frac{|(\bigcap_{l=1}^{k} TR(C_{F_{i,l}})) \cap (\bigcap_{l=1}^{k} TR(C_{F_{j,l}}))|}{\text{MIN}(|\bigcup_{l=1}^{k} TR(C_{F_{i,l}})|, |\bigcup_{l=1}^{k} TR(C_{F_{j,l}})|)} \geq \chi \quad (9)$$

In the definition above, χ designates the *minimum commonness* between two clusters or between two combined features. Finding the correct value for χ is complicated since it heavily depends on the data set. Our experience indicates that $\chi \geq 50\%$ works well in most cases.

Combined features are generated using an Apriori-like algorithm. That is, length-$(k+1)$ features are generated from length-k features by checking the condition in Definition 9. The detailed algorithm is omitted due to lack of space.

6. A CLASSIFICATION STRATEGY

Now, a possible classification strategy is presented to use the features generated by *TraClass*. Here, we do not claim that this strategy is the best among all others, but it is just a reasonable strategy.

Our feature generation framework has good usability for the following reasons. First, it is neutral to classifiers. That is, it can be used for any classifiers, including decision trees, rule-based classifiers, and support vector machines. Second, it can team up with other feature generation frameworks. Trajectories may have multiple numerical attributes in addition to a sequence of locations. Examples include date, time-of-day, and speed. Such numerical attributes are represented as a single value, an interval, or a sequence of values per trajectory. We can use an existing framework [14] for numerical attributes together with our framework for the trajectorial attribute. Handling numerical attributes is beyond the scope of this paper.

After features are generated from a trajectory database, each trajectory is mapped into a feature vector in the feature space. Notice that this mapping should be performed for both the training set and the test set although features are discovered using only the training set. Each entry of a feature vector corresponds to a feature—a region-based cluster, a trajectory-based cluster, or a numerical feature. The i-th entry of a feature vector is equal to the frequency that the i-th feature occurs in the trajectory. When part of a trajectory is contained in a region-based cluster or is close to a trajectory-based one, the corresponding entry of the feature vector is increased by one.

In our study, a classifier is built using the support vector machine (SVM) [18]. This design decision stems from two characteristics of the feature vectors generated. First, they are *high-dimensional* since many features are generated from a trajectory database. Second, they are *sparse* since each trajectory has only a few of these features. The SVM is well-suited for such high-dimensional and sparse data [9].

7. EXPERIMENTAL EVALUATION

7.1 Experimental Setting

We use three real trajectory data sets: the animal movement data set[4], the vessel navigation data set[5], and the hurricane track data set[6].

The animal movement data set has been generated by the Starkey project. The x and y coordinates are extracted for experiments. We use the animal movements observed in June 1995. This data set is divided into *three* classes by species: elk, deer, and cattle, as shown in Figure 15. The numbers of trajectories (points) are 38 (7117), 30 (4333), and 34 (3540), respectively. 20% of trajectories are randomly selected for the test set.

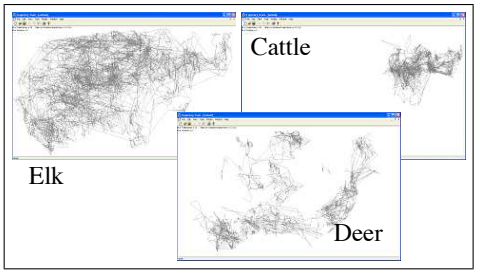


Figure 15: Three classes in the animal data.

The vessel navigation data set has been generated by the MUSE project. The latitude and longitude are extracted for experiments. This data set is divided into *two* classes by vessel: Point Lobos and Point Sur. A larger data set is generated by perturbing the original data set: every trajectory is duplicated with a small ($\leq \pm 5\%$) random value added to the original value. The numbers of trajectories (points) are 600 (65500) and 550 (125750), respectively. 20% of trajectories are randomly selected for the test set.

[4]http://www.fs.fed.us/pnw/starkey/data/tables/
[5]http://www.mbari.org/MUSE/platforms/ships.htm
[6]http://weather.unisys.com/hurricane/atlantic/

Table 2: The summary of the classification results.

Data Set	Animal		Vessel		Hurricane	
Version	TB-ONLY	RB-TB	TB-ONLY	RB-TB	TB-ONLY	RB-TB
Accuracy (%)	50.0	83.3	84.4	98.2	65.4	73.1
Training Time (msec)	3542	2406	44683	22902	331	317
Prediction Time (msec)	104	98	722	608	48	46

The hurricane track data set is called *Best Track*. The latitude and longitude are extracted for experiments. We use the Atlantic hurricanes for the years 1950 through 2006. The Saffir-Simpson scale classifies hurricanes into categories 1∼5 by intensity. A high category number indicates a high intensity. Categories 2 and 3 are chosen for *two* classes. The numbers of trajectories (points) are 61 (2459) and 72 (3126), respectively. 20% of trajectories are randomly selected for the test set.

Classification accuracy, training time, and prediction time are measured for the three data sets. *Classification accuracy* is the ratio of the number of test trajectories correctly classified to the total number of test trajectories. To show the effect of region-based clustering, we test a simple version that performs only trajectory-based clustering. *TB-ONLY* means the simple version, and *RB-TB* the full version that performs the two types of clustering. TB-ONLY is expected to be no worse than earlier methods since it discovers not only sub-trajectory features but also whole-trajectory features by cluster-link generation.

All experiments are conducted on a Pentium-4 3.0 GHz PC with 1 GBytes of main memory, running on Windows XP. Everything is implemented in C++ using Microsoft Visual Studio 2005. Besides, LIBSVM [3] with a linear kernel is used to build a classifier, and the parameters of LIBSVM are set to the default values.

7.2 Results for Real Data

Table 2 summarizes the classification results for the three data sets. In the snapshot images, thin gray lines represent trajectories, thick color lines trajectory-based clusters, and thick color rectangles region-based clusters. Here, a color represents a class.

7.2.1 Results for Animal Movement Data

Figure 16 shows the features generated using $\varepsilon = [29, 39]$, $MinLns = 5$ for elk; $\varepsilon = [44, 54]$, $MinLns = 4$ for deer; and $\varepsilon = [19, 29]$, $MinLns = 5$ for cattle. Ten region-based and 37 trajectory-based clusters are discovered. This result coincides with Figure 15. Many region-based clusters of elk are found along the elk's movements, except where the elk's movements overlap the deer's or the cattle's, *i.e.*, the center and upper-right regions. Trajectory-based clusters of deer are located in the lower-middle and middle-right regions, and those of cattle in the upper-right region.

In Table 2, the accuracy of RB-TB is shown to be much higher than that of TB-ONLY. This indeed demonstrates the advantage of performing the two types of clustering, which we propose in this paper. Moreover, a detailed examination discloses that the test trajectories misclassified are all located in the regions where two classes overlap.

Another interesting observation is that the training time of RB-TB is shorter than that of TB-ONLY, which means region-based clustering can improve efficiency. In Figure 16, many trajectory partitions are covered by region-based clusters and are not provided to the next step of finding trajectory-based clusters. Notice that trajectory-based clustering is more expensive than region-based clustering since the former takes account of movement patterns. Thus, the more trajectory partitions are covered by region-based clusters, the shorter the training time becomes.

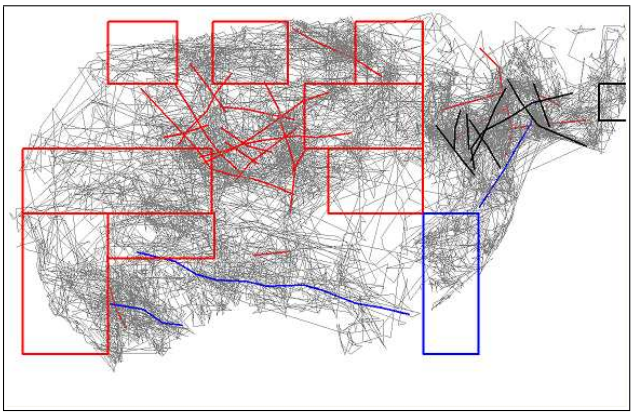


Figure 16: Features for the animal data *(Red: Elk, Blue: Deer, and Black: Cattle).*

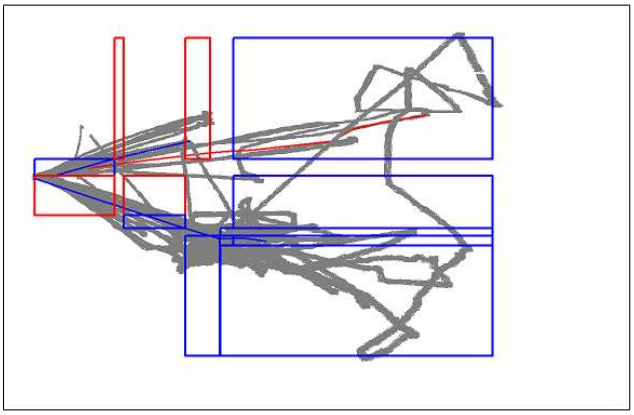


Figure 17: Features for the vessel data *(Red: Point Lobos and Blue: Point Sur).*

7.2.2 Results for Vessel Navigation Data

Figure 17 shows the features generated using $\varepsilon = [24, 34]$, $MinLns = 73$ for Point Lobos and $\varepsilon = [27, 37]$, $MinLns = 76$ for Point Sur. Twelve region-based and three trajectory-based clusters are discovered. RB-TB achieves high accuracy and efficiency since a large proportion of trajectory partitions are covered by region-based clusters. In Table 2, the accuracy of RB-TB is 98.2%, and the training time of RB-TB is only a half of that of TB-ONLY. Besides, see Appendix B for the result of sensitivity analysis.

7.2.3 Results for Hurricane Track Data

Figure 18 shows the features generated using $\varepsilon = [38, 48]$, $MinLns = 3$ for category 2 and $\varepsilon = [42, 52]$, $MinLns = 4$ for category 3. One region-based and 15 trajectory-based clusters are discovered. Classification accuracy for the hurricane track data set is relatively low (TB-ONLY = 65.4% and RB-TB = 73.1%) in comparison with the previous two experiments, because the trajectory of a hurricane is not strongly correlated with its intensity. Nevertheless, two interesting observations can be made from Figure 18. First, a region-based cluster of category 3 appears in far seas, which is natural because stronger hurricanes tend to go further than weaker ones. Second, an obvious trajectory-based cluster of category 3 appears in the lower region. The hurricanes forming the cluster are those that entered the Gulf of Mexico and thus stayed longer at sea before landfall than others. They are likely to get strong because hurricanes gain energy from the evaporation of warm ocean water.

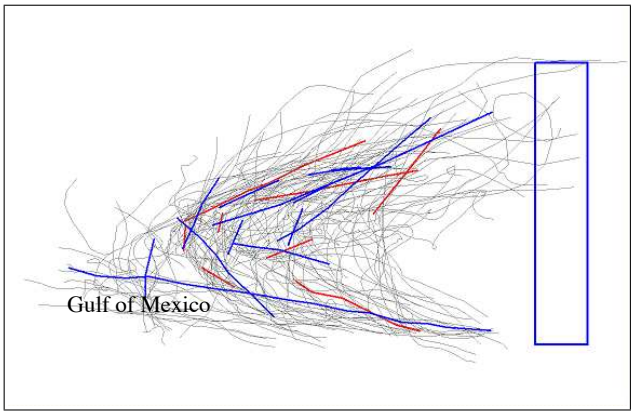


Figure 18: Features for the hurricane data *(Red: Category 2 and Blue: Category 3).*

7.3 Results for Synthetic Data

Due to space limitations, on classification with synthetic data sets we present only two figures: one for accuracy and one for efficiency.

Figure 19 shows classification accuracy as the ratio of region-based clusters is varied when the numbers of training and test trajectories are each 1000. To achieve this, the ratio of the trajectories that are located within homogeneous regions is varied from 0.1 to 0.6. Those trajectories are randomly generated so that common movement patterns do not exist. The result is exactly what we expect. The accuracy of RB-TB is kept very high above 95% regardless of the ratio, whereas that of TB-ONLY deteriorates as the ratio increases. The high accuracy of RB-TB is achieved by the collaboration between the two types of clustering.

Figure 20 shows training time as the number of trajectories is varied when the ratio of region-based clusters is 0.2. Each trajectory has 100 points on average. RB-TB is shown to be scalable since its training time increases almost linearly. The training time of TB-ONLY grows more rapidly than that of RB-TB since trajectory-based clustering is more expensive than region-based clustering. This result indicates another benefit of the collaborative clustering.

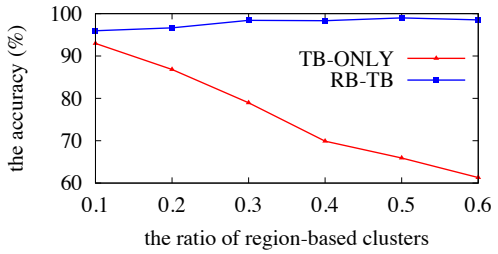


Figure 19: Effect of the degree of homogeneity.

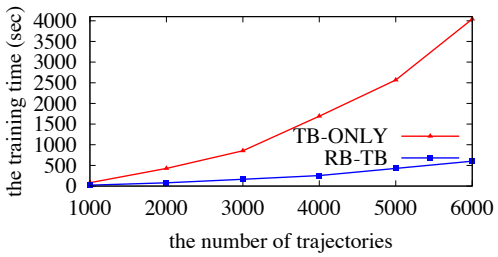


Figure 20: Effect of the data size.

8. RELATED WORK

Trajectory classification has been widely used in the field of pattern recognition such as speech, handwriting, signature, and gesture recognition. Most of proposed methods employ the hidden Markov model (HMM) and use whole trajectories for classification. Bashir *et al.* [1] have presented a framework of classifying human motion trajectories, which uses the HMM with a mixture of Gaussians. A few methods partition trajectories, but the purpose of their partitioning is just to approximate or smooth trajectories before using the HMM [1, 6]. That is, sub-trajectories are not used as separate features. Even though the HMM has been successfully used for modeling this kind of trajectories, it is not suitable for modeling the trajectories in our applications since the Markov property may not hold. For example, in a vessel monitoring application, a certain class may be characterized by whether vessels visit specific two harbors, and the paths between them are not relevant. In this case, the probability of a state between the two harbors is not dependent on that of its immediately previous state.

Trajectory classification has been an active research topic in the fields of bioengineering and video surveillance. Many of proposed methods employ the neural network such as the self-organizing map (SOM) and use whole trajectories for classification. Sbalzarini *et al.* [16] have compared various machine learning techniques used for classifying biological motion trajectories. Owens and Hunter [15] have proposed a method of detecting suspicious behaviors of pedestrians. Here, each trajectory is encoded to a feature vector using its summary information (*e.g.*, the maximum speed). If a trajectory is very complicated, some valuable information could be lost due to this encoding.

Time-series classification is also related to trajectory classification. Time series, however, are very different from trajectories in our applications: they are typically sequences of real values rather than (x, y) points. Besides, time series are compared as a whole for classification. The 1-nearest neighbor classifier using dynamic time warping (DTW) is known to be the best method [20, 21]. Many techniques have been

developed to reduce the overhead of calculating the DTW distance between two time series [21].

Moving-object anomaly detection is more closely related to trajectory classification. Li *et al.* [14] have proposed an anomaly detection method based on motifs (*i.e.*, trajectory features). Their method of extracting motifs, however, is somewhat straightforward. A sliding window of length ω is used to process trajectories, and then, all bounding boxes are clustered to k representative patterns by the k-means algorithm. In fact, their work is focused on incorporating other attributes of trajectories (*e.g.*, date, time-of-day, speed, or proximity to other objects such as landmarks) into classification rather than just extracting motifs. Such attributes are added to the feature space based on their discriminative power. On the other hand, our work is focused on extracting effective trajectory features. Their technique can be adopted in our framework to integrate our trajectory features with other kinds of features.

9. CONCLUSIONS

A novel and comprehensive feature generation framework for trajectories has been proposed in this paper, which performs hierarchical region-based and trajectory-based clustering after trajectory partitioning. Its primary advantage is the high classification accuracy owing to the collaboration between the two types of clustering.

Extensive experiments have been conducted using real and synthetic data sets. Our framework, by visual inspection, has been verified to generate high-quality region-based and trajectory-based clusters. Most importantly, the classification results have demonstrated that performing both types of clustering improves classification accuracy as well as classification efficiency.

Overall, we believe that we have provided a new paradigm in trajectory classification. Various real-world applications, *e.g.*, vessel classification, can benefit from our framework. There are many challenging issues such as integration with numerical-feature generation frameworks, and we are currently investigating into detailed issues as a further study.

10. REFERENCES

[1] F. I. Bashir, A. A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using hidden Markov models. *IEEE Trans. on Image Processing*, 16(7):1912–1919, 2007.

[2] J. N. Cape, J. Methven, and L. E. Hudson. The use of trajectory cluster analysis to interpret trace gas measurements at Mace Head, Ireland. *Atmospheric Environment*, 34(22):3651–3663, 2000.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[4] J. Chen, M. K. Leung, and Y. Gao. Noisy logo recognition using line segment Hausdorff distance. *Pattern Recognition*, 36(4):943–955, 2003.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, Aug. 1996.

[6] R. Fraile and S. J. Maybank. Vehicle trajectory approximation and classification. In *Proc. British Machine Vision Conf.*, pages 832–840, Southampton, UK, Sept. 1998.

[7] H. Greidanus and N. Kourti. Findings of the DECLIMS project—Detection and classification of marine traffic from space. In *Proc. Advances in SAR Oceanography from Envisat and ERS Missions*, Frascati, Italy, Jan. 2006.

[8] P. D. Grünwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, second edition, 2006.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proc. 24th Int'l Conf. on Data Engineering*, pages 140–149, Cancun, Mexico, Apr. 2008.

[12] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proc. 2007 ACM SIGMOD Int'l Conf. on Management of Data*, pages 593–604, Beijing, China, June 2007.

[13] T. C. M. Lee. An introduction to coding theory and the two-part minimum description length principle. *International Statistical Review*, 69(2):169–183, 2001.

[14] X. Li, J. Han, S. Kim, and H. Gonzalez. ROAM: Rule- and motif-based anomaly detection in massive moving object data sets. In *Proc. 7th SIAM Int'l Conf. on Data Mining*, Minneapolis, Minnesota, Apr. 2007.

[15] J. Owens and A. Hunter. Application of the self-organizing map to trajectory classification. In *Proc. 3rd IEEE Int'l Workshop on Visual Surveillance*, pages 77–83, Dublin, Ireland, July 2000.

[16] I. F. Sbalzariniy, J. Theriot, and P. Koumoutsakos. Machine learning for biological trajectory classification applications. In *Proc. 2002 Summer Program, Center for Turbulence Research*, pages 305–316, Aug. 2002.

[17] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[18] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[19] K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. In *Proc. 25th Int'l Conf. on Very Large Data Bases*, pages 363–374, Edinburgh, Scotland, Sept. 1999.

[20] L. Wei and E. J. Keogh. Semi-supervised time series classification. In *Proc. 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 748–753, Philadelphia, Pennsylvania, Aug. 2006.

[21] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proc. 23rd Int'l Conf. on Machine Learning*, pages 1033–1040, Pittsburgh, Pennsylvania, June 2006.

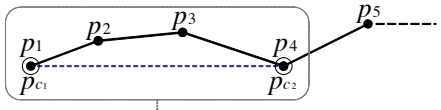
APPENDIX

A. TRAJECTORY PARTITIONING

The trajectory partitioning algorithm in the partition-and-group framework [12] relies on the minimum description length (MDL) principle. A set of trajectory partitions corresponds to H, and a trajectory to D. Thus, finding a good partitioning translates to finding the best hypothesis using the MDL principle. Figure 21 shows our formulation of $L(H)$ and $L(D|H)$. $L(H)$ is formulated by Eq. (10), which represents the sum of the length of a trajectory partition. $L(D|H)$ is formulated by Eq. (11), which represents the sum of the difference between a trajectory and its trajectory partition. Only two components $d_\perp$ and d_θ of the distance function in Section 5.2.1 are considered to measure the difference.

$$L(H) = \sum_{j=1}^{par_i - 1} \log_2(len(p_{c_j} p_{c_{j+1}})) \quad (10)$$

$$L(D|H) = \sum_{j=1}^{par_i - 1} \sum_{k=c_j}^{c_{j+1}-1} \{\log_2(d_\perp(p_{c_j}p_{c_{j+1}}, p_k p_{k+1})) + \log_2(d_\theta(p_{c_j}p_{c_{j+1}}, p_k p_{k+1}))\} \quad (11)$$



$L(H) = \log_2(len(p_1p_4))$

$L(D|H) = \log_2(d_\perp(p_1p_4, p_1p_2) + d_\perp(p_1p_4, p_2p_3) + d_\perp(p_1p_4, p_3p_4)) + \log_2(d_\theta(p_1p_4, p_1p_2) + d_\theta(p_1p_4, p_2p_3) + d_\theta(p_1p_4, p_3p_4))$

Figure 21: Formulation of the MDL cost.

The selected partitioning is the one that minimizes $L(H)+L(D|H)$. An approximate algorithm has been developed since the cost of finding the exact solution is prohibitive. The key idea of our approximation is to regard the set of local optima as the global optimum. Let $MDL_{par}(p_i, p_j)$ denote the MDL cost ($= L(H) + L(D|H)$) of a trajectory between p_i and p_j ($i < j$) when assuming that p_i and p_j are only partitioning points. Let $MDL_{nopar}(p_i, p_j)$ denote the MDL cost when assuming that there is no partitioning point between p_i and p_j, *i.e.*, when preserving the original trajectory. Then, a local optimum is the *longest* trajectory partition p_ip_j that satisfies $MDL_{par}(p_i, p_k) \leq MDL_{nopar}(p_i, p_k)$ for every k such that $i < k \leq j$. The algorithm based on this idea is shown in Figure 22.

B. SENSITIVITY ANALYSIS

Classification accuracy is measured for the vessel navigation data set while varying two parameter values: ε_{vend} (the largest value in the ε-interval) and $MinLns$. Region-based clustering is related to only $MinLns$, and trajectory-based clustering to both parameters.

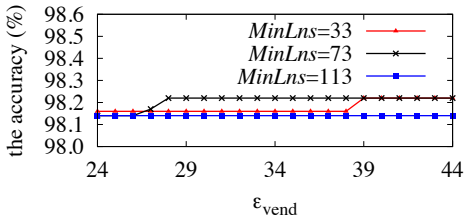
Figure 23 shows the results of RB-TB, and Figure 24 those of TB-ONLY. In Figures 23(a) and 24(a), the parameter values for the former class are varied, with those for the latter fixed to the values estimated by the heuristic. In Figures 23(b) and 24(b), the parameter values for the latter class are varied, with those for the former fixed. The estimated value of ε_{vend} is placed in the middle of the X-axis.

Algorithm **Approximate Trajectory Partitioning**

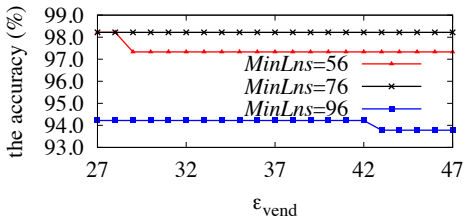
INPUT: A trajectory $TR_i = p_1p_2p_3 \cdots p_j \cdots p_{len_i}$
OUTPUT: A set $\mathcal{P}_i$ of partitioning points
ALGORITHM:
01: Add p_1 into the set $\mathcal{P}_i$; /* the starting point */
02: $startIndex := 1$, $length := 1$;
03: **while** $startIndex + length \leq len_i$ **do**
04: $currIndex := startIndex + length$;
05: $cost_{par} := MDL_{par}(p_{startIndex}, p_{currIndex})$;
06: $cost_{nopar} := MDL_{nopar}(p_{startIndex}, p_{currIndex})$;
/* Check if partitioning at the current point makes the MDL cost larger than not partitioning */
07: **if** $cost_{par} > cost_{nopar}$ **then**
/* Partition at the previous point */
08: Add $p_{currIndex-1}$ into the set $\mathcal{P}_i$;
09: $startIndex := currIndex - 1$, $length := 1$;
10: **else**
11: $length := length + 1$;
12: Add p_{len_i} into the set $\mathcal{P}_i$; /* the ending point */

Figure 22: An approximate algorithm for partitioning a trajectory [12].

A good sign from Figure 23 is that the classification accuracy of RB-TB is shown to not be sensitive to parameter values when region-based clusters cover a large proportion of trajectory partitions as in Figure 17. Moreover, the estimated value achieves the maximum accuracy. Therefore, we expect that the region-based and trajectory-based dual framework can reduce the burden of parameter tuning, which is a very desirable property.



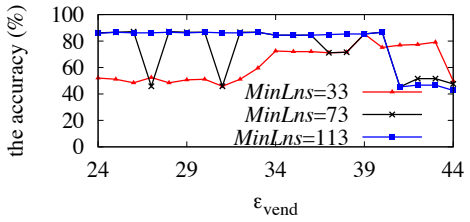
(a) Varying the parameters for *Point Lobos*.



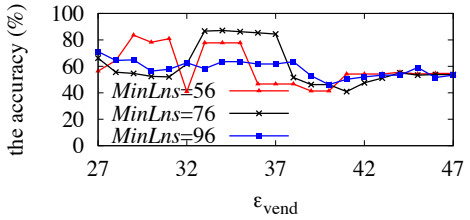
(b) Varying the parameters for *Point Sur*.

Figure 23: Parameter sensitivity of RB-TB for the vessel data.

In contrast, the classification accuracy of TB-ONLY is shown to be rather sensitive to parameter values. However, a reasonably large range of parameter values around the estimated one provide classification accuracy very close to the maximum. These results clearly show the effectiveness of our heuristic.



(a) Varying the parameters for *Point Lobos*.



(b) Varying the parameters for *Point Sur*.

Figure 24: Parameter sensitivity of TB-ONLY for the vessel data.

C. SEQUENCE CLASSIFICATION

One might argue that sequence classification can be applied after converting each trajectory into a sequence. More specifically, the domain space is partitioned into a set of grids, and a trajectory is represented using the labels of the grids traversed by the trajectory. The sequence-conversion method is intuitive and simple, but if a 2-dimensional trajectory were converted into a sequence, the exact information of location and direction could be lost. Furthermore, it is hard to determine the optimal granularity of a grid.

In fact, the method proposed by Fraile and Maybank [6] is based on sequence classification. A trajectory is first divided into overlapping segments. For each segment, the trajectory of a vehicle is approximated by a smooth function and then assigned to one of four categories: ahead, left, right, or stop. In this way, the list of segments is reduced to a string of symbols drawn from the set $\{a, l, r, s\}$. The string of symbols is classified using the HMM. As discussed in Section 1, their method might not be able to discover region and sub-trajectory features, whereas *TraClass* can.