ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets*

Xiaolei Li Jiawei Han Sangkyum Kim Hector Gonzalez

University of Illinois at Urbana-Champaign, IL, USA

Abstract

With recent advances in sensory and mobile computing technology, enormous amounts of data about moving objects are being collected. One important application with such data is *automated identification of suspicious movements*. Due to the sheer volume of data associated with moving objects, it is challenging to develop a method that can efficiently and effectively detect anomalies. The problem is exacerbated by the fact that anomalies may occur at arbitrary levels of abstraction and be associated with multiple granularity of spatiotemporal features.

In this study, we propose a new framework named ROAM (Rule- and Motif-based Anomaly Detection in Moving Objects). In ROAM, object trajectories are expressed using discrete pattern fragments called *motifs*. Associated features are extracted to form a hierarchical feature space, which facilitates a multi-resolution view of the data. We also develop a general-purpose, rule-based classifier which explores the structured feature space and learns effective rules at multiple levels of granularity.

We implemented ROAM and tested its components under a variety of conditions. Our experiments show that the system is efficient and effective at detecting abnormal moving objects.

1 Introduction

In recent years, the collection of historical or real-time data on *moving objects* is quickly becoming a ubiquitous task. With the help of GPS devices, RFID sensors, RADAR, satellites, and other technologies, mobile objects of all sizes, whether it be a tiny cellphone or a giant ocean liner, can be easily tracked across the globe. As a result, many new applications are being developed. Examples include indexing and querying of moving objects over static or continuous queries [6, 9, 18, 20, 8], similarity search between moving objects, and continuous clustering of moving objects [7, 11].

Example. At any one time, there are approximately 160,000 vessels traveling in the United States' waters. They include military cruisers, private yachts, commercial liners, and so on. It is unrealistic to manually examine each of these vessels and identify the suspicious ones. Thus, it is possible, and highly desirable, to create automated tools that can evaluate the behavior of all maritime vessels and develop situational awareness on the abnormal ones.

Example 1 shows one particular problem which is of interest in homeland security and surveillance. It is *automated detection of suspicious or anomalous moving objects*. This is our focus in this paper.

An outlier is, in general, viewed as "*an observation (or a set of observations) which appears to be inconsistent with the remainder of that set of data* [2]." However, the term "inconsistent" has many far-reaching implications. The decision is often subjective and depends heavily on the context. Outliers are also rare in the population, which makes search harder.

Though outlier detection has been studied in many contexts [2, 13], the moving objects domain [8] poses unique challenges. Additionally, problems such as indexing [18, 20], clustering [12, 7, 11], anomaly detection [16, 15] have been studied in moving objects domain as well. However, they focus almost exclusively on the trajectories. In practice, trajectories are associated with non-spatiotemporal features and such associations are often more valuable for analysis. In this paper, we take a different approach by constructing a multi-dimensional feature space oriented on segmented trajectories. This allows us to analyze complex relationships between multiple features associated with different granularities and dimensions in each trajectory.

There are in general two mechanisms for anomaly

* The work was supported in part by The Boeing Company and the U.S. National Science Foundation NSF IIS-05-13678/06-42771 and NSF BDI-05-15813. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

detection: *classification*, which relies on labeled training data sets, and *clustering*, which performs automated grouping without the aid of training data. Although both are interesting methods for mining moving object outliers, classification often leads to stronger mining results with the help of training data. Therefore, our focus will be on constructing a classification model.

1.1 Problem Definition The problem of anomaly detection in moving object data is defined as follows. The input data is a set of **labeled trajectories**: $\mathcal{D} = \{(t_1, c_1), (t_2, c_2), \ldots\}$, where t_i is a trajectory and c_i is the associated class label. A *trajectory*[1] is a sequence of spatiotemporal records of a moving object, *e.g.*, GPS records. Each record has the geographic location as well as a timestamp, and records can be made at arbitrary time intervals. The set of possible class labels is $\mathcal{C} = \{c^1, c^2, \ldots\}$. In simple anomaly detection, there could just be two classes: c^{normal} and $c^{abnormal}$.

The goal of the problem is to learn a function f which maps trajectories to class labels: $f(t) \rightarrow c \in \mathcal{C}$. f should be consistent with $\mathcal{D}$ as well as future trajectories not in $\mathcal{D}$. In other words, we want to learn a model which can classify trajectories as being normal or abnormal.

1.2 Contributions In this paper, we propose a framework, called ROAM (Rule- and Motif-based Anomaly Detection in Moving Objects), for the problem of anomaly detection. Compared to related work in classification or clustering of moving objects, ROAM incorporates a fuller feature space and examines more than just trajectories. At a high level, ROAM presents three novel features.

1. **Motif-based feature space**: Instead of modeling whole trajectories, we partition them into fragments (*motifs*) and construct a multi-dimensional feature space oriented on the motifs with associated attributes.

2. **Automated hierarchy extraction**: By examining the patterns in the trajectories, we automatically derive hierarchies in the feature space. This yields a multi-resolution view of the data.

3. **Hierarchical rule-based classifier**: We develop a rule-based classifier which explores the hierarchical feature space and finds the effective regions for classification.

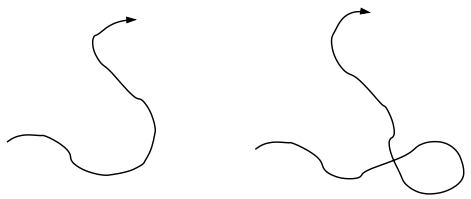
[1]Trajectory in this paper is just data and does not imply path prediction.

The rest of the paper is organized as follows. Section 2 presents some key insights in ROAM. In Section 3, we introduce the overall framework. Experimental results are shown in Section 4. Section 5 addresses the related work. And we conclude the study in Section 6.

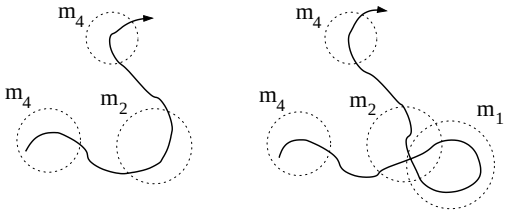
2 Key Insights

There have been some prior work in the area of trajectory prediction [16, 15]. Markov models or other sequential models can model a single trajectory and predict its future behavior. However, when used in the context of a large population with many different distributions, such approaches may not be effective.

Example. Consider the two trajectories in Fig. 1(a). They have similar shapes except the one on the right has an extra loop. The impact of this additional loop depends on the task, but one would remark that the other portions are remarkably similar.



(a) Two similar trajectories. The loop in the right trajectory is difficult to handle in holistic approaches.



(b) Same two trajectories after motif extraction. The right trajectory has an extra m_1.

Figure 1: Motif representation

This example presents some problems for holistic models. It is difficult to represent the semantics of "mostly the same with the exception of an extra loop" using distance metrics between models. Local differences could either dominate the metric or be drowned out by the rest of trajectory. Furthermore, it is difficult to capture thousands or tens of thousands of trajectories in a single model. While a single object or a small set may have clear patterns, a large population (such as in real-world anomaly detection) presents a wide range of patterns across all granularities of time and space signals.

2.1 Motif-based Feature Space In this paper, we propose that semantic analysis should be based on a rich feature space constructed using trajectory fragments. In ROAM, raw trajectories are partitioned into fragments. These fragments are overlaid on top of each other and the common patterns become what we call **motifs**. Motifs are represented by a tuple (motif expression) which includes additional spatiotemporal attributes that may be helpful in analysis. The set of motif expressions observed then forms a feature space in which the original trajectories are placed. Using such a feature space, we can leverage algorithms in machine learning and data mining to learn complex associations between trajectory fragments and also other important information.

A *motif* is a prototypical movement pattern. Examples include *right turn*, *u-turn*, and *loop*. One could view them as parallels to gene expressions in DNA sequences or entity mentions in text. Fig. 1(b) shows the motifs in Fig. 1(a) as drawn by the dotted circles. In this form, the two trajectories now have much in common: They share one m_2 and two m_4's, and differ in one m_1 (*i.e.*, *loop* motif).

2.2 Multi-Resolution Feature Hierarchies Another observation we make is raw recordings and semantic analysis often occur at different spatiotemporal granularities. While time recordings may be made at the minute or second level, analysis is usually more sensible on the hour level or even higher. The same scenario applies to the location measure. One might record at the meter level but analyze at the city block or district level.

By using a more general representation, fewer distinct measure values are used and the analysis task could become easier. In addition, it would improve human readability. If the final classification model produces human readable results (as ROAM does), having high level features not only reduces the size of the results but also increases their understandability.

Sometimes, these hierarchies are readily available, as in the case of time. With other features, however, it may not be obvious. In ROAM, we use a clustering-based technique to automatically extract hierarchies based on the behavior of the trajectories. Given such hierarchies, it is still hard to know a priori which levels will work the best for classification so we let ROAM adjust dynamically.

3 Framework

Figure 2 shows the ROAM (Rule- and Motif-based Anomaly Detection of Moving Objects) framework. Square boxes are computation modules, round boxes are data sources, and arrows show the flow of data. There are three computation modules in ROAM: Motif Extractor, Feature Generator, and Hierarchical Rule-based Classifier. Data flows through them in that sequence.

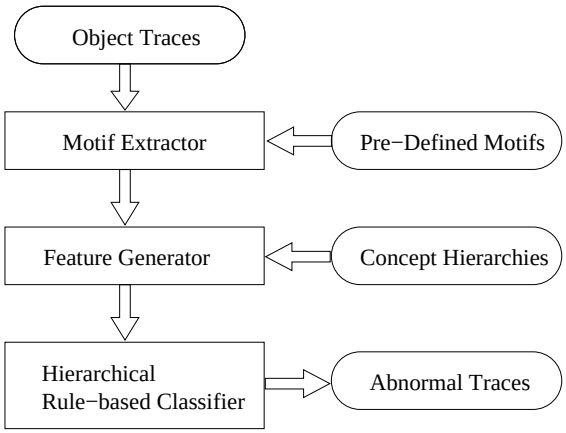


Figure 2: ROAM Framework

3.1 Motif Extractor The first computation module in ROAM is the Motif Extractor. A motif is a prototypical movement pattern. ROAM use a sliding window technique to process the trajectories. All windows are overlaid on top of each other and clustering is used to group them into representative sets. These representative ones then form the set of interesting *motifs* in $\mathcal{D}$.

Given a trajectory, we slide a window of length ω across it. ω could be defined with respect to time or distance. If it is time, then different speeds (and thus distance traveled) would result in different motifs. If it is distance, then speed variances would be normalized. In our experiments, we used time since speed was relatively stable though more complex data might require more complex normalization methods. For each resultant window w, we compute the vector from the first point in w to the last point in w. The width of the vector is then expanded to accommodate all other points within w; this bounding box allow us to smooth over noises in the trajectory.

All bounding boxes are overlaid on top of each other. And using the Euclidean distance, we cluster them to find the representative patterns. The resultant cluster centers then define the set of motifs. In ROAM, a motif is represented just like a window: a vector with a bounding box around it. Depending on the task, other variables may be kept as well. Once the motifs are set, we then go through $\mathcal{D}$ again using the same sliding windows. This time, a window w in a trajectory is *similar* to a motif m if $||w - m|| \leq \epsilon$. And if a particular window is similar to a motif, we say that

motif is "expressed" in the trajectory.

A natural question to raise is how to set ω. A too small of a value could miss motifs by dividing them into indistinguishable pieces and a too large of a value could bundle motifs together and lose discriminative power. Fortunately, it turns out that most reasonable values will perform just fine. As we will show empirically in Section 4, classification accuracy is fairly robust with regards to different ω values.

Given a trajectory, the motif extractor returns the sequence of **motif expressions** found in the trajectory. Each motif expression has the form

$$(m_i, t_{start}, t_{end}, l_{start}, l_{end}) \tag{3.1}$$

where m_i is the motif, t_{start} and t_{end} are the starting and ending times, and l_{start} and l_{end} are the starting and ending locations. The complete sequence is known as the **motif trajectory** of the original trajectory.

3.1.1 Motif Expression Attributes The form of motif expression shown in Eq. (3.1) is only the first step in full motif expression extraction. Additional information on when, where, and how the motif was expressed is needed. Take Fig. 3 as an example. There are two objects moving in an area with the same trajectories; however, the left one is near an important landmark. This extra piece of information (*i.e.*, proximity to landmark) can be crucial in decision making. If we also knew that the left object was moving slowly during the middle of the night, the combination of all such information is telling in anomaly detection.

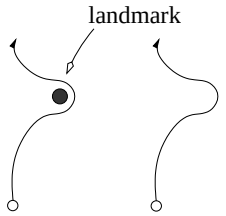


Figure 3: Two objects moving with the same trajectory.

For each motif expression, we introduce a set of **attributes** in addition to the simple time and location ones in Eq. (3.1). Some examples include `duration`, `top_speed`, `avg_speed`, `radius`, and `general_location`. Some of these attributes can be derived easily from the time and location attributes, *e.g.*, `avg_speed` = *path-distance*$(l_{start}, l_{end}) \div (t_{end} - t_{start})$. Others may require a more sophisticated motif extractor.

Let there be A such attributes: $\{a_1, a_2, \ldots, a_A\}$. We now represent each motif expression as follows,

$$(m_i, v_1, v_2, \ldots, v_A) \tag{3.2}$$

where m_i is the motif and v_i is the value of attribute a_i. Note that a_i may be continuous or even multi-dimensional.

3.2 Feature Generator Once the motif expressions have been extracted, semantic analysis can begin. One could try the following naïve classification scheme. For each distinct (*motif, attribute, attribute value*) combination we see in the trajectory data, we map it to a feature. For example, (right-turn, `speed`, $11mph$) would map to a feature and (right-turn, `speed`, $12mph$) would map to another feature. Formally, $\forall\ i, j, k\ (m_i, a_j, v_k) \leftrightarrow f_x \in \mathcal{F}$ where $\mathcal{F}$ is the resulting feature space. We then use the following classifier.

Algorithm 1. (Flat-Classifier)

1. *Transform the motif-trajectories into vectors in the $\mathcal{F}$ feature space. Suppose $f_x \leftrightarrow (m_i, a_j, v_k)$. Then the x^{th} component of the vector has the value of the number of times motif i's attribute j expressed value k in the trajectory.*
2. *Feed the feature space and the data points as input into a learning machine.*

This particular transformation from trajectories to a feature space is complete. Every motif attribute value is preserved as a feature and the frequencies of their expressions are preserved as the feature values. However, it is ineffective by the following observations. First, a large number of distinct motif attribute values leads directly to a high dimensional feature space. Specifically, suppose there are M motifs, A attributes, and each attribute has V distinct possible values. The instance space is then $\mathbb{N}^{MAV}$. Second, the high granularity or continuous motif attribute values make generalization difficult. Because these distinct values are transformed to distinct features, generalization becomes essentially impossible. Learning on a feature that is at `10:31am` will have no bearing on a feature that is at `10:32am`.

3.2.1 Feature Generalization In order to overcome the difficulties in the Flat-Classifier, generalization in the feature space is needed. For example, (*right-turn*, `time`, $2am$), (*right-turn*, `time`, $3am$), and (*right-turn*, `time`, $4am$) features could be generalized into one feature: (*right-turn*, `time`, *early_morning*). This not only reduces the dimensionality of the feature space but also helps the learning machine through feature extraction.

Recall that each feature has the form (m_i, a_j, v_k), where each attribute a_j is either numerical (1D) or spatiotemporal (2D or 3D). We assume that each a_j has a distance metric defined on its values. Thus, features having the same m_i and a_j values (*i.e.*, $(m_i, a_j, *)$)

can be compared with a formal distance metric. For example, (*right-turn*, `time`, $2am$) is more similar to (*right-turn*, `time`, $2:02am$) than (*right-turn*, `text`, $6pm$). But, it does not make sense to compare features with different m_i or a_j values (*e.g.*, (*right-turn*, `time`, $2am$) is not comparable to (*u-turn*, `speed`, $10mph$)).

We partition the features in $\mathcal{F}$ into sets with distinct (m_i, a_j) values. If there are M motifs and A attributes, there are $M \times A$ disjoint sets. We propose to generalize the features in each (m_i, a_j) set into a smaller set. Further, this new set will be hierarchical where appropriate. This will be the task of the **Feature Generator** in the ROAM framework. Specifically, it will

1. discretize or cluster continuous or high granularity motif attribute values.

2. form a hierarchy over the attribute values, which in turn offers a multi-resolution view of the data.

We will treat each (m_i, a_j) space independently. Since the attribute values can have different forms (*e.g.*, numerical values, 2D spatial locations), we will use different methods where appropriate. We explain them in detail in the following two sections.

3.2.2 Spatial Attributes Attributes such as `location` are spatial points in a 2D or 3D space. In such scenarios, we use a hierarchical "micro-clustering" technique similar to BIRCH [26] to discover prototypical patterns. Features are inserted into a tree-based data structure where nodes represent micro-clusters. A micro-cluster is a small, tightly grouped neighborhood, and features belong to the same micro-cluster only when they are closely related. A tree of these micro-clusters represents a concept hierarchy of the attribute.

Take the `location` attribute as an example. A micro-cluster may only include features which are within a few meters of each other. During insertion of features into the tree, each micro-cluster has a maximum radius parameter. If a feature cannot fit inside a micro-cluster, a new micro-cluster is created. The tree also had a maximum branching factor so insertions and rotations occur like a typical B-tree. After all features have been inserted into the tree, the leaf nodes form the set of micro-clusters. Each micro-cluster can be viewed as a meta data point that represents similar features. We then feed the set of micro-clusters into a hierarchical agglomerative clustering algorithm to construct the final hierarchy.

The final clustering tree is hierarchical in the following sense: any node in the tree contains summarized information for all data points in that node's subtree. For example, the root contains a summary of the entire tree. The summary information is sufficient for the calculation of the centroid and the radius of the points. The reason we choose a BIRCH-like algorithm in our system is two-fold. First, it performs micro-clustering, which fits our needs better. Second, building the CF tree is time and space efficient ($O(n)$). More properties are described in [26].

3.2.3 Numerical Attributes Attributes such as `time` and `avg_speed` are numerical. Usually, in the presence of continuous attributes, *discretization* is performed. Doing so has many advantages. First, it makes the learning problem easier. A decision tree, for example, would have fewer splits to consider. A discrete feature allows better generalization. Second, it makes human readability easier. For instance, it is much easier to understand the feature value of `1pm-2pm` as opposed to reading all the distinct values between `1pm` and `2pm`.

Discretization techniques [17] can be split into two main groups: unsupervised and supervised. Since we have labeled data, supervised algorithms are more appropriate. There is an abundant number of methods available for this; most of them would function just fine here. An additional requirement we have is a hierarchy over the resultant discrete values. While most discretization methods do not have this property, we can easily add it by performing hierarchical agglomerative clustering as a post-processing step.

Since spatial attributes are a generalization of numerical attributes, we use the same clustering methods in our implementation of ROAM for both types of attributes. Clustering in one-dimensional data still provides meaningful groupings based on behavior.

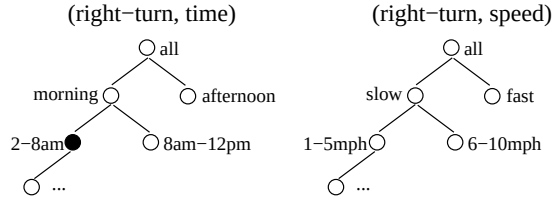


Figure 4: Two sample motif-attribute hierarchies

3.2.4 Multi-Resolution View After building hierarchies in each of the (m_i, a_j) spaces, the overall feature space is now structured as a set of hierarchies. Fig. 4 shows a partial illustration. In it, there are two **motif-attribute hierarchies**: (*right-turn*, `time`) and (*right-turn*, `speed`). Each node corresponds to a micro-cluster feature discovered in $\mathcal{F}$. For example, the black node in Fig. 4 represents all right-turns taken between 2 and

8am. High level nodes in the hierarchies correspond to high level features and low level nodes correspond to low level features. By choosing different subsets of the nodes, a user can create distinctly different views of the data. For example, suppose one only used level one features in Fig. 4 (*i.e.*, "morning", "slow", etc). This generates a very rough view of the data and with only four features. On the other hand, choosing the leaf nodes in Fig. 4 generates a detailed view but with many more features.

As mentioned previously, concept hierarchies may already exist for some attributes (*e.g.*, time). In such cases, one may just choose to use them to construct the motif-attribute hierarchy. However, in other cases or sometimes in place of the existing hierarchies, one could use automated techniques in ROAM to construct the hierarchies. This has the distinct advantage that the hierarchies are built based on the behavior of the data. As a result, more features could be dedicated to the dense regions and fewer features to the sparse regions. Clustering and discretization techniques can adjust dynamically based on the data and could facilitate more effective analysis.

3.3 Classification Let $\mathcal{F}'$ be the set of all nodes in the motif-attribute hierarchies. $\mathcal{F}'$ is the largest feature space where all resolutions are included. Though this feature space is "complete", it is unlikely to be the best one for classification. It creates a high dimensional feature space; this makes learning slow and possibly ineffective. On the other hand, suppose we choose only the root level nodes in all the motif-attribute hierarchies. That is, only the "all" nodes in Fig. 4. The problem with this feature space is that the features are too general to be useful. What we seek is something in between those two extremes.

To this end, we propose a rule-based classification method: CHIP (Classification using Hierarchical Prediction Rules). We chose a rule-based learning algorithm for several reasons. First and foremost, it produces human-readable results. This is useful in practice. Second, it is efficient. CHIP is $O(N)$ with respect to either the number of examples or the number of features. Other classifiers such as Naïve Bayes or SVM are $O(N^2)$ with respect to one. The problems we are dealing could be large and high dimensional. Lastly, the classification problem is unbalanced: the abnormal class has few training examples. In such contexts, rule-based learner have been shown to be effective [5].

3.3.1 Intuitions Before formally describing CHIP, we give some intuitions. CHIP iteratively and greedily searches for the best available rule until all positive examples are covered. In addition, CHIP tries to use high-level features whenever possible. For example, suppose all ships that move at location X between the hours of *12pm* and *5pm* are normal. Then a single rule using the *afternoon* feature will suffice. Using this principle has several benefits. First, the feature space is kept to a small size. This speeds up learning and keeps the problem tractable. Second, features are kept high level whenever possible. This produces rules which are general and easily understood by a human. Third, it avoids the problem of over-fitting.

In machine learning research, the study of feature space simplification or generalization is known as *feature selection* [10]. Given a set features, choose a subset which will perform better (in terms of efficiency and/or accuracy) in the learning task. A typical approach scores each feature and iteratively inserts or removes them. In our setting, however, we have something that is different than the standard setting: there are hierarchical structures over the features. Thus, selection should be a little smarter.

With this in mind, we propose a top-down search in the feature hierarchies. We start with an initial high level feature space and try to describe the data (in the rules sense). If these features produce accurate rules, we are satisfied. But if at some point we find that a more specific feature will produce a better rule, we *expand* the existing feature space to include that specific feature. This process repeats until all the training data is sufficiently covered.

3.3.2 CHIP We introduce some definitions first. CHIP learns a set of rules, much like FOIL [19] and CPAR [25]. A single rule r has the conjunctive form of:

$$l_1 \wedge l_2 \wedge \ldots \wedge l_n \rightarrow c$$

where each l_i is a literal (or predicate) of the form $(feature = value)$ and c is the class label. An example is "covered" by r if all the literals in r are satisfied in the example. Next, recall that $\mathcal{F}'$ is the complete set of features. For any feature f in $\mathcal{F}'$, let $Exp(f)$ return the set of f's children in $\mathcal{F}'$'s hierarchy. For example, $Exp(\texttt{morning}) = \{\texttt{2-8am}, \texttt{8am-12pm}\}$. At any time, CHIP uses a subset of the features in $\mathcal{F}'$. Let $\mathcal{F}_C$ be this set.

A rule is learned one literal at a time. Literals are selected according to a weighted version of $Foil\_Gain$ [19], which is based on the positive and negative coverage of the rule before and after adding the literal. Let p_0 and n_0 be the number of positive and negative examples covered by rule r without literal l. Let p_1 and n_1 be the number of positive and negative examples covered

by rule $r \wedge l$. $Foil\_Gain(l, r)$ is then defined as

$$p_1 \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

The weighted version of $Foil\_Gain$ [25] allows previously covered positive examples to be used again but just weighs them down. This adjusts the p and n values appropriately in the above equation.

In the previous section, we gave the intuitive notion of discovering that a more specific feature will perform better than a current feature. Here, we formalize this notion in the function $Exp\_Gain(f, r)$ where f is a feature and r is a rule. It is defined as

$$Exp\_Gain(f, r) = \max_{(l, f_i) \forall l, f_i \in Exp(f)} Foil\_Gain(l, r)$$

The Exp_Gain (expansion gain) of a feature is the maximum $Foil\_Gain$ achieved by any literal in any of its child features. It is defined with respect to a non-empty rule similar to $Foil\_Gain$. We chose this function because it allows sensible direct numerical comparisons between $Foil\_Gain$ and Exp_Gain.

ALGORITHM 2. (CHIP)

Input: *(1) Training set $\mathcal{D} = P \cup N$, where P and N are the positive and negative examples. (2) Initial feature set $\mathcal{F}_\mathsf{C} \in \mathcal{F}'$.*

Output: *Set of classification rules R.*

Method:

```
1.  while not all of P is covered
2.      initialize new rule r
3.      while true
4.          find literal l with highest Foil_Gain(l, r)
5.          find feature f with highest Exp_Gain(f, r)
6.          if both gains < min_gain then break
7.          if Foil_Gain(l, r) > β · Exp_Gain(f, r) then
8.              add l to r
9.          else
10.             add feature f to F_C
11.     add r to R
12. return R
```

Discussion CHIP starts with all examples uncovered and iteratively searches for the best rule to cover the positive examples. The search is greedy and halts when enough positive examples are covered. Rules are learned one literal at a time, choosing them based on $Foil\_Gain$ (line 4). In line 5, the Exp_Gain of each feature is calculated. If the better gain is $Foil\_Gain$, the literal is added to the current rule (line 8). Otherwise, the feature space is expanded (line 10) and the process repeats.

Complexity CHIP has running time of $O(nSR)$ where n is the number of examples, S is the size of the used feature space, and R is the number of learned rules. In our implementation, we collapsed examples (trajectories) which appear the same into meta-examples. Thus, with a high initial feature space, n can be quite small if the data is skewed. S can also be small initially since there are only a few high level features. As the algorithm executes, both n and S will increase with feature expansion. This is another reason to avoid careless expansion.

4 Experiments

In this section, we show our framework's performance in a variety of settings. We conduct our experiments using both real and generated data to show efficiency and effectiveness. For data generation, we used GSTD [21] (which generates raw trajectories) and also our own data generator (which generates motif-trajectories). The latter allows us to test some parts of the framework independently of others. Efficiency experiments were run on an Intel Pentium 4 2.6GHz machine with 1.5GB of memory. The Motif Extractor was written in Python and the rest was written in C++ and compiled with GCC.

Each dataset consists of two classes: normal and abnormal. In GSTD, we achieve this by generating two datasets with slightly different parameters. In our own generator, the base data is motif-trajectories. A set of motif-expression seeds are initialized in the model and a Gaussian mixture model is used to create randomness. We generate the abnormal class by mixing "abnormal" motifs with a background model that is shared between both the normal and abnormal classes.

There are two parameters which controls CHIP. One is the starting level in the motif-attribute hierarchy. Level 0 denotes the root level. The other is β, the feature expansion weight. These two parameters are indicated as ROAM(*starting_level*, β). Finally, since the number of abnormal examples is small, we used the standard F1 metric instead of accuracy. F1[2] is a harmonic mean of recall and precision and better reflects the effectiveness of the classifier. An F1 score of 100 indicates 100% recall and precision. F1 scores were the result of 10-fold cross validation. Experiments were run 5 times to get an average.

4.1 Real Data We obtained real ship navigational data from the Monterey Bay Aquarium Research Institute (MBARI[3]). Under the MUSE project, several

[2] $F1 = (2 \times recall \times precision) \div (recall + precision)$

[3] `http://www.mbari.org/MUSE/platforms/ships.htm`

ships traveled in ocean waters near Northern California to conduct various aquatic experiments. The ships' navigational data, which includes time, longitude, and latitude, were recorded. Depending on the ship, the sampling rate varied from 10 seconds to a few minutes; the end result are fairly continuous paths. Figure 5 shows a typical path of a vessel named Point Sur.

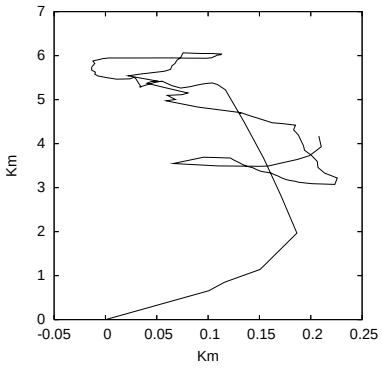
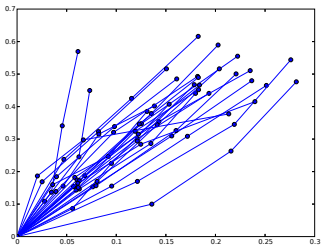


Figure 5: Sample path of ship Point Sur from 16:00 to 24:00 on 8/23/00 starting at point $(0, 0)$.

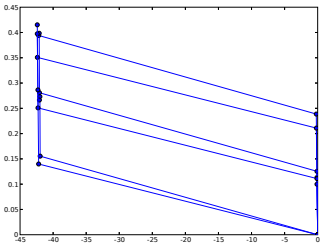
We collected data from two different ships (namely Point Sur and Los Lobos) and assigned different class labels to them. The two ships carried out different tasks and thus naturally had different movement patterns. There was a total of 23 paths (11 of one, 12 of another), each with 1500 to 4000 points. Using ROAM, we extracted 40 motifs, constructed features, and tried to recover the class labels using CHIP. Figure 6 shows two sets of trajectory segments that were marked as motifs 10 and 14. Motif 10 is a simple straight trajectory towards the northeastern corner. Motif 14 is a 3-part move of going north, northwest, and then north again.

Motifs were extracted from a window of approximately 3 minutes, and had two additional attributes. One is the distance traveled, which indicates speed, and the other is the general Euclidean distance to the stored motif. We did not include the time-of-day attribute since the two ships had regular but different schedules and including them would make the problem too easy. Motif-attribute hierarchies (branching factor of 4) were also generated, which ranged from 2 levels deep to 7 levels deep.

An issue raised before was the setting of ω, the width of the window to extract motifs. Figure 7 shows the effect on classification as ω increases from 4 minutes to 60 minutes on the MBARI data. As shown, accuracy with too small or too large of a window is poor, but in the intermediate, it is relatively stable. Thus, we believe that as long as the window is reasonable, performance should not be affected too much. Another



(a) Extracted motif 10



(b) Extracted motif 14

Figure 6: Extracted motifs from MBARI data.

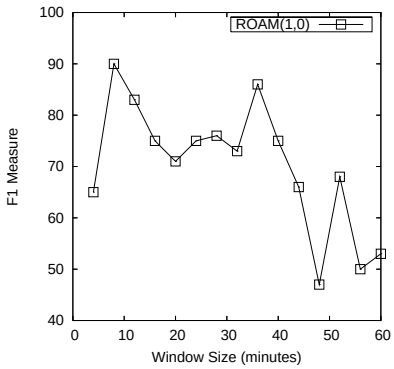


Figure 7: Effect of ω on classification accuracy.

issue is how *many* motifs to extract. This was set to 40 in Figure 7, and Figure 8 shows the effect as that number changes from 5 to 60. The curve shows that we were able to achieve 100% classification accuracy with 10 and 15 motifs. And as the number increases, accuracy decreases but not too drastically. In general, a reasonable number should not be too difficult to find.

4.2 Synthetic Data While the real data experiments provided some validation of our methods, we were unable to thoroughly test other aspects due to the small dataset size. To combat this, we experimented with synthetic data from GSTD and also our own data generator.

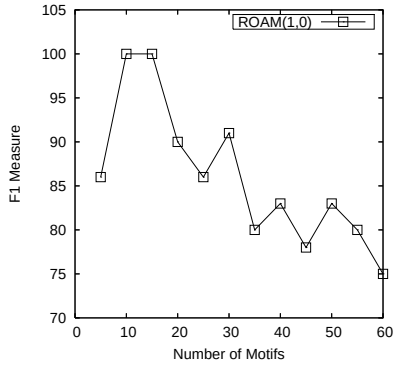


Figure 8: Effect of number of motifs on classification accuracy.

4.2.1 Notation For our own data generator, we use the following notation to denote the parameters used in generation. Each data set's name is in the form of "$N\#B\#M\#A\#S\#L\#$", where N is the number of normal trajectories, B is the number of abnormal ones, M is the number of motifs, A is the number of attributes, S is the standard deviation in the Gaussian mixture distributions, and L is the average length of the trajectory.

4.2.2 Classification Accuracy First, we tested accuracy using GSTD. In GSTD, we generate two different classes of data using Gaussian distributed movement centers. The two models shared the same parameters except the mean of the centers differed by 0.01 (0.50 vs. 0.51). Fig. 9 shows the results as we also varied the variance in the distributions. As expected, accuracy improves as the trajectories differ more from each other. But even at small differences, ROAM was able to distinguish the two classes.

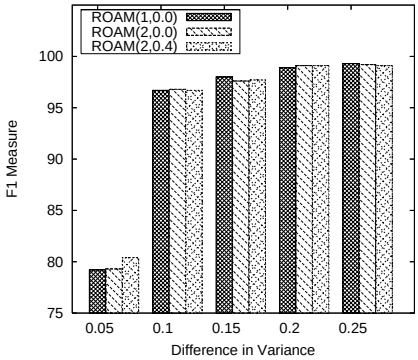


Figure 9: GSTD $N2000B200M30$: Accuracy with respect to difference in variance.

Next, we tested the accuracy using our own data generator. Fig. 10 shows F1 results as the number of motifs in the data increased from 10 to 100 on the y-axis. For comparison, we used SVM[4] (nu-SVC with radial kernel) with the Flat-Classifier as described before and also SVM with level 2 features. The first thing we notice is that SVM with Flat-Classifier is hopeless, as expected. We also observe that ROAM with level 1 features and a little bit expansion is almost as good as SVM with level 2 features. ROAM with level 2 and a little bit expansion is equal to or better than SVM.

We note that the size of the classification feature space is much larger than the number of motifs. For example, when the number of motifs equals 100, the number of level 2 features equals nearly 1200.

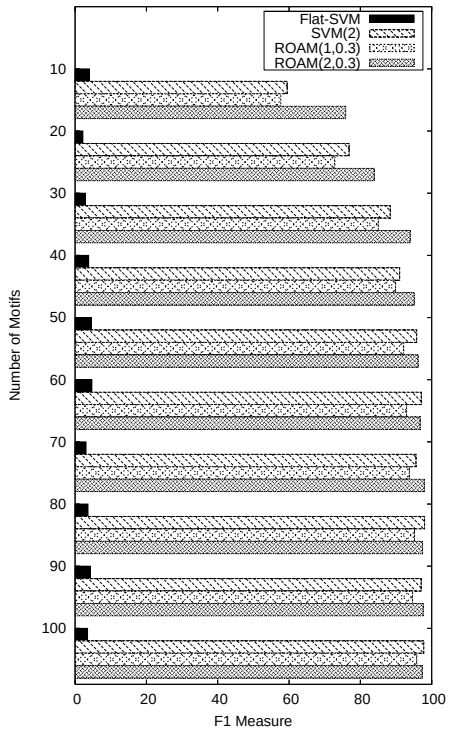


Figure 10: $N4kB200A3S5.0L20$: Accuracy with respect to number of motifs.

Fig. 11 shows F1 as the motif-trajectory length varies. As the length increases, the data gets denser and we observe that SVM's performance deteriorates. However, ROAM with its various configurations were fairly stable. Fig. 12 shows the effect as standard deviation is increases from 5 to 40. As expected, F1 decreases as the values get more spread out. One might have noticed that we have rather large standard

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm

deviation values. This is because the range of values is large (∼1000).

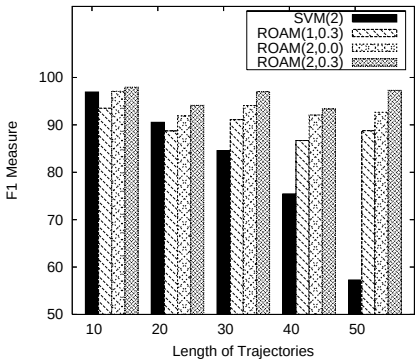


Figure 11: $N4kB200A3S5.0L20$: Accuracy with respect to length of motif-trajectories.

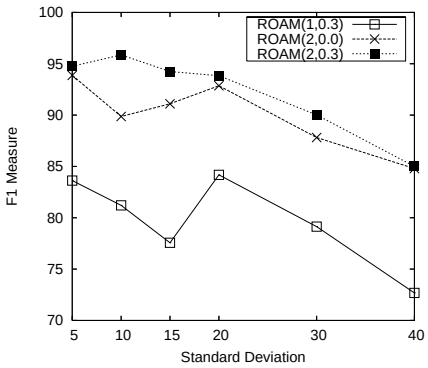


Figure 12: $N5kB100M20A3L20$: Accuracy with respect to standard deviation.

Recall that a larger value of β, the expansion factor, increases the chances that CHIP will expand the feature space during learning. The effect of different β values vary from one dataset to another. Fig. 13 shows a typical result. In the ROAM(1,$*$) curve, ROAM starts with level 1 features and improves significantly with expansion. In the ROAM(2,$*$) curve, F1 is high initially. It improves slightly with some expansion but eventually drops down. This is the effect of over-fitting. In other words, CHIP has expanded too greedily and the feature space has become too specific.

Finally, Fig. 14 compares a general feature space vs. a specific one. One is ROAM(2,0), which is level 2 features with no expansion. The other is ROAM(MAX), which is only the leaf features. We see that ROAM(2,0) is significantly better in accuracy. Furthermore, it is also faster. With 60 motifs, ROAM(2,0) took an average of 84 seconds with 705 features while ROAM(MAX) took

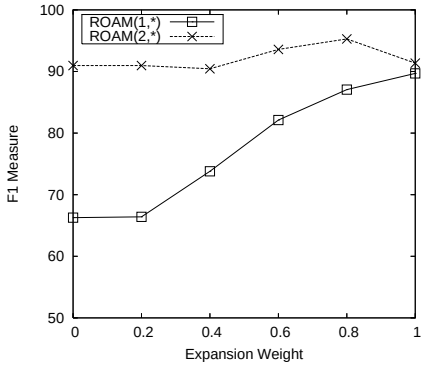


Figure 13: $N500B100M20A3S25L20$: Accuracy with respect to β.

850 seconds with approximately 4350 features.

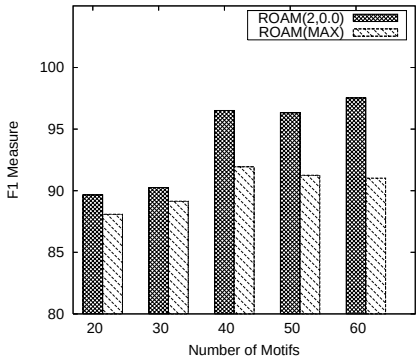


Figure 14: $N15kB500A3S25L20$: Accuracy with respect to number of motifs.

4.2.3 Efficiency With regards to efficiency, we first check sensitivity to the number of trajectories. Fig 15 shows a plot broken down into ROAM's components, note the log scale. As we can see, all components scale nicely with respect to the number of trajectories. The Motif Extractor is the slowest, but it was implemented in Python (10–30 slower than C++) while the other components were in C++.

Another aspect of efficiency is sensitivity to the length of the trajectories. Fig. 16 shows the running time as the length was increased from 10 to 100 in our own data generator. Fig 17 shows a similar experiment using GSTD data. Again, we see a linear increase in running time as trajectory length increased. The reason is that with longer trajectories, there is a linear increase in the number of motif expressions ROAM has to process.

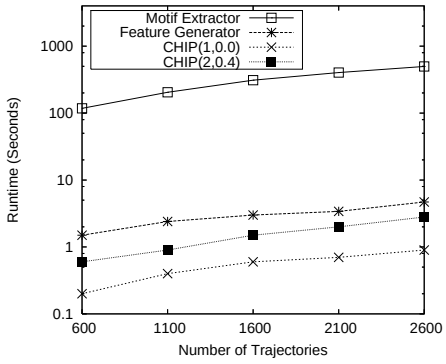


Figure 15: GSTD: $B200M20$: Efficiency with respect to number of trajectories.

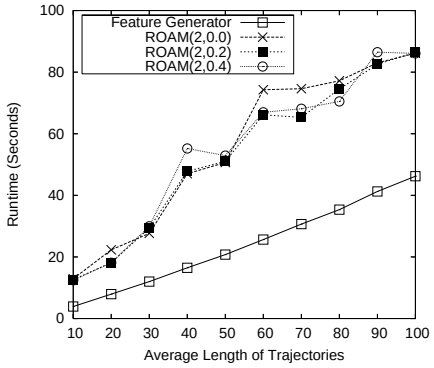


Figure 16: $N20000B1000M20A3S10.0$: Efficiency with respect to length of motif-trajectories.

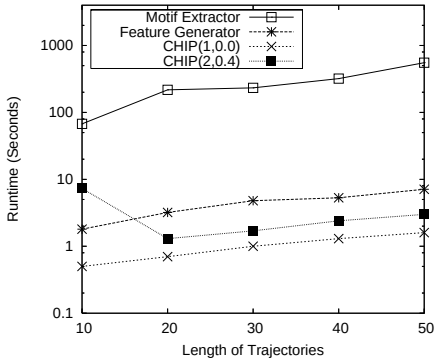


Figure 17: GSTD: $N2000B100M10$: Efficiency with respect to length of trajectories.

5 Related Work

Prior work in moving object databases (MOD) has addressed problems similar to ours. Discrete data model of moving objects was introduced in [6] and used subsequently in [16, 3, 15]. However, to our knowledge, ROAM is the first work to represent trajectories in a feature space oriented on the discrete units (fragments). Prior work used fragments as a dimensionality-reduction technique and still retained a trajectory model.

In [16, 15], the authors construct models to predict trajectories as well as recognize anomalies. However, both works assume the existence of a single trajectory model. This works well when there is only a few objects ([16] experimented with one object and [15] experimented with three). In such cases, the object(s) have clear individual patterns and could be captured in a compact model. However with anomaly detection in a large population, it is unclear whether such approaches will work. Within the population, there is a very large variation of trajectories and anomalies could occur in any form. In ROAM, we do not assume the existence of a single or a few global trajectory models. The classifier can learn rules specific to any scenario.

Clustering moving objects [7, 11] and time series [12] as well as classification of time series [24, 23] are also related to this work. However, most of them focus analysis on the raw trajectories using techniques such as dynamic time warping. In ROAM, the interactions between the trajectories and non-spatiotemporal attributes play a crucial role that is usually ignored in previous work. In real world cases, such interactions often convey the most useful information.

In addition to specific problems, representation in MOD [8] is a core issue. In [9], abstract data types are added to a DBMS to model the geometries. In [6], a discrete data model is proposed. Trajectories are decomposed into "slices" where each slice is represented by a simple function. In comparison to our work, these slices are simpler motifs and do not translate to features. Related to representation, indexing and query processing [18, 20] are also key MOD issues. However, they focus analysis on the raw spatial and trajectory data and query processing. For example, discovering which moving objects are physically located within a certain query window. In ROAM, we focus on the semantic problem of anomaly detection which requires higher level analysis.

Work in time series and traditional anomaly detection also touches on our problem. [1] presents techniques which can query for shapes in time series, and [4] automatically discovers motifs in time series. Algorithms in this area are helpful and could be applied in ROAM, but our framework does more than just motif discovery. It builds a hierarchical feature space using the motifs and performs high level feature analysis. Traditional outlier detection [13] is closely tied to our problem. However, they are only concerned with fixed

data points in 2D space. We focus on moving object data.

In data mining, there is work which focuses on finding frequent patterns related to moving objects. [22] mines sequential patterns in spatial locations, and [14] mines co-location association rules. However, such studies are often at a higher semantic level than ROAM. That is, they capture associations between locations but ignore patterns in raw trajectories (as well as their associations).

6 Discussion and Conclusion

In this paper, we have proposed the ROAM framework for the problem of anomaly detection in massive moving object data sets. With advances in tracking technology and increases in the need for better security, automated solutions for detecting abnormal behavior in moving objects such as ships, planes, vehicles, *etc.* are needed more than ever. However, this is a difficult problem since patterns of movement linked with the environment are complex. In ROAM, we use a novel motif-based feature space representation with automatically derived hierarchies. Combined with a rules-based classification model that explores the hierarchies, ROAM is shown to be both effective and efficient in our testing.

With ROAM, we can also start thinking about other data types in the moving object domain. Objects that move in transportation networks have structured data that can be easily translated to ROAM. In addition, we have been thinking about indexing, query processing, clustering, local anomaly detection, and frequent pattern analysis in the ROAM framework. A motif feature based approach changes the model of many problems and could generate many interesting solutions.

References

[1] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. In *VLDB'95*.

[2] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 1994.

[3] H. Cao and O. Wolfson. Nonmaterialized motion information in transport networks. In *ICDT'05*.

[4] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD'03*.

[5] François Denis. Pac learning from positive statistical queries. In *ALT'98*.

[6] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *SIGMOD'00*.

[7] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *KDD'99*.

[8] G. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[9] R. H. Güting, M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. In *TODS'00*.

[10] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. In *Journal of Machine Learning Research*, 2003.

[11] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD'05*.

[12] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD'98*.

[13] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98*.

[14] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *SSD'95*.

[15] V. Kostov, J. Ozawa, M. Yoshioka, and T. Kudoh. Travel destination prediction using frequent crossing pattern from driving history. In *ITSC'05*.

[16] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *AAAI'04*.

[17] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6:393–423, 2002.

[18] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *VLDB'00*.

[19] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *ECML'93*.

[20] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD'00*.

[21] Y. Theodoridis, J.R.O. Silva, and M.A. Nascimento. On the generation of spatiotemporal datasets. In *SSD'99*.

[22] I. Tsoukatos and D. Gunopulos. Efficient mining of spatiotemporal patterns. In *SSTD'01*.

[23] L. Wei and E. Keogh. Semi-Supervised Time Series Classification. In *KDD'06*.

[24] X. Xi, E. Keogh, C. Shelton, and L. Wei. Fast Time Series Classification Using Numerosity Reduction. In *ICML'06*.

[25] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *SDM'03*.

[26] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD'96*.