Motion-Alert: Automatic Anomaly Detection in Massive Moving Objects⋆

Xiaolei Li   Jiawei Han   Sangkyum Kim

Department of Computer Science, University of Illinois at Urbana-Champaign

Abstract. With recent advances in sensory and mobile computing technology, enormous amounts of data about *moving objects* are being collected. With such data, it becomes possible to automatically identify suspicious behavior in object movements. *Anomaly detection* in massive moving objects has many important applications, especially in surveillance, law enforcement, and homeland security.

Due to the sheer volume of spatiotemporal and non-spatial data (such as weather and object type) associated with moving objects, it is challenging to develop a method that can efficiently and effectively detect anomalies of object movements in complex scenarios. The problem is further complicated by the fact that anomalies may occur at various levels of abstraction and be associated with different time and location granularities. In this paper, we analyze the problem of anomaly detection in moving objects and propose an efficient and scalable classification method, called *motion-classifier*, which proceeds in the following three steps.

1. Object movement features, called *motifs*, are extracted from the object paths. Each path consists of a sequence of motif expressions, associated with the values related to time and location.
2. To discover anomalies in object movements, motif-based generalization is performed that clusters similar object movement fragments and generalizes the movements based on the associated motifs.
3. With motif-based generalization, objects are put into a multi-level feature space and are classified by a classifier that can handle high-dimensional feature spaces.

We implemented the above method as one of the core components in our moving-object anomaly detection system, *motion-alert*. Our experiments show that the system is more accurate than traditional classification techniques.

1   Introduction

In recent years, gathering data on *moving objects*, *i.e.*, the objects that change their spatial locations with time, has become an easy and common task. The

tracking of mobile objects, whether it be a tiny cellphone or a giant ocean liner, is becoming increasingly accessible with embedded GPS devices and other sensors. Such enormous amount of data on moving objects poses great challenges on effective and scalable analysis of such data and exploration of its applications. One application which is of particular interest in homeland security is the *detection of suspicious or anomalous moving objects*: i.e., automatic identification of the abnormal or suspicious moving objects from among a massive set of object movements.

Example 1 There are a large number of vessels sailing near American coasts. It is unrealistic to manually trace such enormous number of moving objects and identify suspicious ones. Although there could exist a good number of previous case studies or training examples on suspicious vessels, it is highly desirable to develop automated tools that can evaluate the behavior of *all* maritime vessels and flag the suspicious ones. This will allow human agents to focus their monitoring more efficiently and accurately. ∎

In this paper, we take vessels in Example 1 as typical type of moving objects and study how to develop scalable and effective methods for automated detection of anomalous moving objects. We believe such methods can be easily extended to other applications. In general, there are two general mechanisms for anomaly detection: *classification*, which relies on training data sets, and *clustering*, which performs automated grouping without using training sets. Here we focus our study on *classification*, because it is not hard to find a good set of training examples, and with good quality training data, classification often leads to higher accuracy than clustering-based methods in anomaly detection.

There are several major challenges in anomaly detection of moving objects. First, tracking moving objects can generate an enormous amount of high precision and complex data, consisting of both spatiotemporal and non-spatial information. For example, the time and location of a vessel might be recorded every few seconds, and non-spatial information such as the vessel's weight, speed, shape, and paint-color may be included in the recordings. Second, there exist substantial complexities of possible abnormal behavior, which may occur at arbitrary levels of abstraction and be associated with different time and location granularities. The massive amount of data and complexity of abnormal patterns make efficient and effective anomaly detection very challenging.

In this study, we systematically study this problem and propose an effective and scalable classification method for detecting anomalous behavior in moving objects. It features the following components.

1. **Motif-based representation**: Instead of viewing the movement path of an object as a sequence of low-level spatiotemporal points, we view it as a sequence of movement motifs.
2. **Motif-oriented feature space transformation**: The movement paths are transformed into a feature space that is oriented on the movement motif expressions.

	Non-Spatio-Temporal Features		Path	Class
	Size	... Type		
1	Cruiser	... Military	$\langle\ldots\rangle$	−
2	Cruiser	... Commercial	$\langle\ldots\rangle$	−
3	Cruiser	... Civilian	$\langle\ldots\rangle$	+
4	Sailboat	... Commercial	$\langle\ldots\rangle$	−
5	Sailboat	... Civilian	$\langle\ldots\rangle$	−

Table 1. Original input data

3. **Clustering-based motif feature extraction**: Generalized motif features are extracted from the movement paths. We perform micro-clustering to extract higher level features.
4. **High-dimensional, generalized motif-based classification**: A classifier is learned using the extracted generalized motif features.

The rest of the paper is organized as follows. In Section 2, we describe the representation of moving object data based on movement motifs. In Section 3, we describe the extraction of higher level features from the motif expressions. Experimental results are shown in Section 4. We discuss some related works in Section 5 and conclude the study in Section 6.

2 Movement Motifs

We assume that the input data consists of a set of labeled *movement paths*, $\mathcal{P} = \{p_1, p_2, \ldots\}$, where each path is a sequence of time-related points of an object, and each object is associated with a set of non-spatiotemporal attributes that describe non-motion-related properties. Table 1 shows such a data set extracted from a naval surveillance example, where each ship has non-spatiotemporal attributes, such as "size" and "type", along with the movement path. The "class" column labels the case as either positive (suspicious) or negative (normal).

2.1 Extraction of Movement Motifs

Although the concrete spatiotemporal data could be in the precision of seconds and inches, it is necessary to extract *movement motifs* at certain higher abstraction level in order to perform efficient and semantically meaningful analysis. Consider the two ship movements shown in Fig. 1. They share similar movements except an extra loop in the dashed path. To make semantically meaningful comparisons between moving objects, we propose to extract movement motifs at a proper level for further reasoning and classification.

A **movement motif** is a prototypical movement pattern of an object, often pre-defined by domain experts. Typical examples include *straight line*, *right turn*, *u-turn*, *loop*, *near-an-island*, etc. Let there be M defined motifs: $\{m_1, m_2, \ldots, m_M\}$. A movement path is then transformed to a sequence of motifs with other pieces
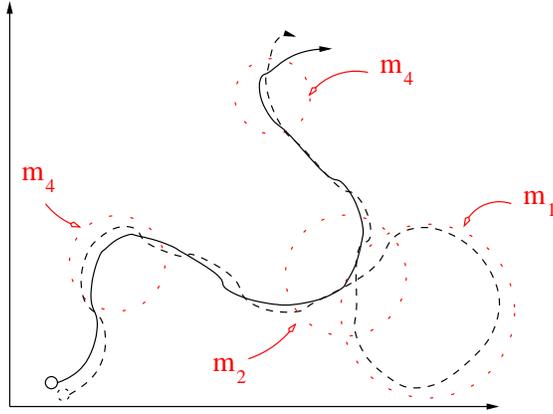
Fig. 1. Extracting motif expressions in raw paths

of information associated. Fig. 1 shows this extraction process. Expressed by a sequence of motifs, the two paths now have much in common: They share one m_2 and two m_4's, and their only difference is an extra m_1 in the dashed path.

We can assume there is a *movement motif extractor* (as preprocessing) that recognizes a pre-defined set of motifs modulo some geometric transformations (*e.g.*, rotation, translation, resizing, etc). In general, each path takes the form,

$$\langle (m_i, t_{start}, t_{end}, l_{start}, l_{end}), (m_j, t_{start}, t_{end}, l_{start}, l_{end}), \ldots \rangle \tag{1}$$

where m_i is an expressed motif, t_{start} and t_{end} are the starting and ending times, and l_{start} and l_{end} are the starting and ending locations. In a single motif expression, $t_{start} < t_{end}$ since each motif must take some non-zero time to execute. In a full path, motifs may be expressed in overlapping times and/or locations. A single motif maybe expressed multiple times within a single movement path.

2.2 Motif-Oriented Feature Space

Recall that the motif extractor is able to extract some preliminary spatiotemporal information about the motif expressions. Using such information, we can derive a set of **attributes**, *e.g.*, `duration`, `avg_speed`, and `generalized_location`. Take Fig. 2 as an example, there are three ships moving in an area that contains an important landmark. The left and right ships have the same movement shapes except that the left one makes its circle around a landmark. This extra piece of information (*i.e.*, general location) combined with the movement motif can be crucial in decision making. If we had the information that the left ship made its movement very late at night and at very slow speeds, the combination of all such features is very telling in anomaly detection.

Let there be A such interesting attributes: $\{a_1, a_2, \ldots, a_A\}$. For each a_i, we map the set of distinct values (*e.g.*, intervals, location coordinates, *etc.*) to a
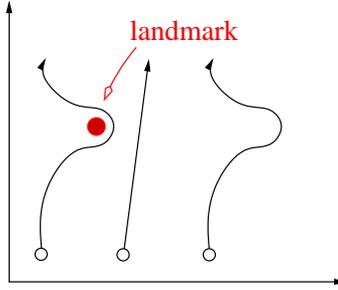
Fig. 2.

	Motif Expressions	Class
1	(Right-Turn, $3am, l_7$), (U-Turn, $4pm, l_2$), (U-Turn, $10pm, l_1$)	−
2	(U-Turn, $10am, l_2$)	−
3	(Left-Turn, $6am, l_7$), (U-Turn, $11am, l_3$), (Right-Turn, $1pm, l_7$), (Right-Turn, $4pm, l_7$)	+
4	(Right-Turn, $1am, l_1$), (U-Turn, $9am, l_1$), (Left-Turn, $3pm, l_3$), (U-Turn, $3pm, l_3$)	−
5	(Right-Turn, $2am, l_1$), (Left-Turn, $9pm, l_3$)	−

Table 2. Motif-oriented database

set of integers between 1 and V, where V is the largest needed value. We now represent each movement path in the following form,

$$\langle (m_i, v_1, v_2, \ldots, v_A), (m_j, v_1, v_2, \ldots, v_A), \ldots \rangle \tag{2}$$

where each tuple is a **motif expression** and v_i is the value of attribute a_i. The set of all possible motif expressions, $(m_i, v_1, v_2, \ldots, v_A)$, plus the non-spatiotemporal features, define the **motif-oriented feature space**.

Table 2 shows a sample transformation from Table 1 to a **motif-oriented database**. For space consideration, we did not include the non-spatiotemporal features. The "Path" feature in Table 1 has been transformed to sets of motif expressions here. In this particular case, we have three motifs: Right-Turn, Left-Turn, and U-Turn. Each motif has two attributes: Time and Location. Notice that this model is still a simplified one for illustration of the major concepts. In some analysis, additional attributes, such as average, maximum, or minimum speed of the vessel, may need to be associated with a motif for successful anomaly detection.

One could try to feed the motif-oriented database into some learning machine. But such attempts may prove to be futile. In the raw database, motif expressions can be recorded with high granularity values, $i.e.$, the number of distinct values for each attribute is very large. And since each distinct expression is mapped to a feature in the motif-oriented feature space, this makes generalization difficult. Take the time-of-the-day attribute as an example. If it is stored at the second level, almost all motif expressions will have different values. Because these different values are stored as different features, generalization becomes essentially

impossible. Learning on a feature that is at `10:30:01am` will have no bearing on a feature that is at `10:30:02am`.

Thus, a different approach is needed to overcome this problem. We propose an extraction process named **Motif Feature Extraction** which will smooth out the high granularity values and extract higher level features. For example, the *exact* distance from a ship to a landmark is unimportant, and it is enough to consider "rather close to it" in comparison with most other ships. We explain this process in detail in the following section.

3   Motif Feature Extraction

The **Motif Feature Extraction** (MFE) process *clusters* the features belonging to each motif attribute and extracts higher level features. Recall that each feature in the motif-oriented feature space is a tuple of the form $(m_i, v_1, v_2, \ldots, v_A)$, where v_j is the value of attribute a_j. For each m_i and a_j combination, MFE independently clusters the values for a_j. For example, suppose we are dealing with the time-of-the-day attribute, MFE will extract representative time concepts based on the data distribution. The result may correspond to a special period in the time of the day, such as *1-3am*, or *late_night*.

These newly found clusters can be seen as higher level features in the data. And thus they will replace the original high granularity values in each a_j. As a result, each feature in the motif-oriented feature space will be a tuple of the form $(m_i, c_1, c_2, \ldots, c_A)$, where c_j is the cluster for attribute a_j. Because the number of clusters will be much smaller than the number of original values for each attribute, the feature space will be greatly reduced. We term this new reduced feature space the **MFE Feature Space**.

In order to cluster the attribute values, we have to define a distance metric for each attribute. In general, a distance metric might not exist for an attribute (*e.g.*, categorical data). Fortunately, because the attributes here are spatio-temporal, distance metrics are naturally defined. For example, attributes related to speed or time are just one-dimensional numerical values. Attributes related to spatial location can be measured with 2D or 3D Euclidean distance metrics.

3.1   Feature Clustering

Now that we have the distance metric for each attribute, we wish to find clusters in the attribute value space. These clusters will replace the original features and form a more compact feature space. We use a hierarchical micro-clustering technique similar to BIRCH [26] to cluster the features. The goal is not to cluster into rather high-level clusters. We only wish to extract some micro-clusters which capture some semantic-level information.

CF Vector   First, we introduce some basic concepts. Given n features (motif-expressions) in the meta-feature space: $\{f_1, f_2, \ldots, f_n\}$, the centroid C and radius

R are defined as follows. Recall that each feature f_i is a A-dimensional vector in the meta-feature space.

$$C = \frac{\sum_{i=1}^{n} f_i}{n} \tag{3}$$

$$R = \left(\frac{\sum_{i=1}^{n} ||f_i - C||^2}{n} \right)^{\frac{1}{2}} \tag{4}$$

Next, we define the **clustering feature** (CF) vector. A single CF vector is a summary of a cluster of features. Given n features in a cluster, the CF vector is a triplet defined as: $CF = (n, LS, SS)$, where n is the number of features in the cluster, LS is the linear sum vector of the n features, *i.e.*, $\sum_{i=1}^{n} f_i$, and SS is the square sum vector of the n features, *i.e.*, $\sum_{i=1}^{n} f_i^2$.

The CF Additivity Theorem [26] shows that if $CF_1 = (n_1, LS_1, SS_1)$ and $CF_2 = (n_2, LS_2, SS_2)$ are the CF vectors of two disjoint clusters, the CF vector of the union is

$$CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2) \tag{5}$$

This additive property is central to the incremental clustering technique in BIRCH. First, it allows the CF vector of a cluster to be calculated incrementally from subsets. And second, the centroid and radius can be calculated from the CF vector.

CF Tree A CF tree is a height-balanced tree with two parameters: branching factor b and radius threshold t. Each non-leaf node consists of at most b entries of the form $(CF_i, child_i)$, where (1) $i = 1, 2, \ldots, b$, (2) $child_i$ is a pointer to its i-th child node, and (3) CF_i is the CF of the cluster represented by $child_i$. A leaf node has only a *leaf entry*, which contains just the CF for that node. The tree is hierarchical in the following sense: the CF at any node contains information for all data points in that node's subtree. Lastly, all leaf entries have to satisfy the threshold t constraint, which restricts the radius of an entry to be less than t. A CF tree is built up dynamically as new data objects are inserted. Insertion is similar to that of a B+-tree.

Properties of CF Tree Building the CF tree is efficient, the cost of constructing a CF tree from N points is $O(N)$. Furthermore, it is designed to fit inside main memory by adjusting b and t dynamically. More properties are described in [26].

3.2 MFE Feature Space

After building the CF tree from the features belonging to a single motif, we take the leaf nodes from the tree. These leaf nodes are the micro-clusters and their centroids are the *extracted features*. They will *replace* the original features to form a new feature space. We term this feature space the **MFE feature space**.

3.3 Classification

After MFE has performed some primitive generalization in the individual dimensions, we employ the use of a classifier to learn more complex generalizations on the database. Since the MFE feature space is fairly high dimensional, we make use of the support vector machine in this learning task.

4 Experiments

In this section, we test our classification system's performance in a variety of settings.

4.1 Data Generation

To systematic study the performance of the method under different data distributions, we generated our own test data sets. Each dataset consists of a set of paths divided into two classes: positive and negative. Both classes use the same background model to generate a sequence of motif expressions. The positive class model has an additional model to insert "abnormal" motif patterns.

The background model is a Gaussian mixture distribution over the space of possible motif expressions. It randomly chooses a set of seeds from this space and generates a Gaussian distribution (independently for each attribute) for each one. During path generation, a length is randomly generated and motif expressions are then randomly generated according to the Gaussian mixture model. Both the positive and negative classes follow this first step.

The positive class has an additional model to generate "abnormal" motif patterns which make it different. Each pattern consists of one or more motif expressions. This additional model is another Gaussian mixture distribution over the motif expression space. During path generation, one or more abnormal motif pattern(s) are inserted into the positive paths.

4.2 Models

For comparison, we constructed a simple baseline model, Plain-SVM. It is an SVM trained on the motif-oriented database without MFE (*i.e.*, Table 2). Our model is named MFE-SVM. The branching factor b of the CF tree is set to 4 in all experiments.

4.3 Experiments

Table 3 shows experimental results in a variety of settings. The first column lists the different datasets. All datasets had 2000 normal paths and 2000 suspicious paths. Each path's average length was 50 motif expressions. Each suspicious path had one additional abnormal motif pattern (randomly generated from the model), which consisted of 2 correlated motif expressions.

The other parameters are shown in the table. Each dataset's name is in the form of "$\#M\#A\#V\#T\#S$", where M is the number of motifs, A is the number of attributes, V is the number of values, T is the number of abnormal patterns in the abnormal model, and S is the standard deviation in the Gaussian mixture distributions. For the MFE-SVM, we tried 2 different numbers for the parameter t, which adjusts the granularity of the CF tree. All SVMs used a radial kernel and all accuracy values were the average of 5 different datasets, each with 10-fold cross validation.

Dataset	Plain-SVM	MFE-SVM $(t = 40)$	MFE-SVM $(t = 150)$
$10M3A1000V20T2.0S$	52%	74%	74%
$10M3A1000V20T1.0S$	57%	72%	73%
$10M3A1000V20T0.5S$	80%	73%	73%
$20M5A1000V20T2.0S$	50%	92%	72%
$20M5A1000V20T1.0S$	51%	93%	71%
$20M5A1000V20T0.5S$	58%	96%	72%
$20M10A1000V20T2.0S$	51%	95%	97%
$20M10A1000V20T1.0S$	51%	97%	97%
$20M10A1000V20T0.5S$	57%	99%	96%
$40M10A1000V20T2.0S$	50%	96%	96%
$40M10A1000V20T1.0S$	53%	99%	96%
$40M10A1000V20T0.5S$	58%	99%	98%
$40M10A1000V50T2.0S$	50%	89%	85%
$40M10A1000V50T1.0S$	51%	95%	92%
$40M10A1000V50T0.5S$	62%	95%	93%
$Average$	55.4%	90.9%	85.6%

Table 3. Classification accuracy: on plain datasets vs. on motif-oriented datasets

As the experiments show, classification accuracies with the MFE features were much better than the plain method. This is easy to justify since generalization in the plain case was essentially impossible (*i.e.*, the SVM functioned as a simple rote learner). In the MFE features, similar spatio-temporal motif expressions were clustered together to allow the SVM to generalize.

Clustering Parameter In Table 3, we see that when the t parameter was adjusted higher, classification accuracies decreased. Recall t controls the size of the micro-clusters in the CF trees (*i.e.*, bigger t means bigger clusters). In terms of motifs, a larger t means rougher granularities, or higher generalizations, in the attribute measurements. For example, using *late_night* vs. {*1-2am, 2-3am, ...*} to represent the time measurement. Also, note that setting t to 0 is equivalent to using the Plain-SVM.
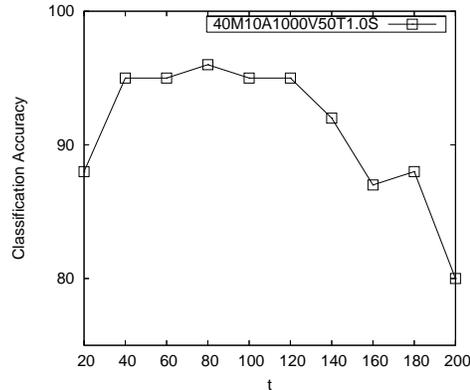
Fig. 3. The effect of the t parameter on classification.

Fig. 3 shows the effects of t on classification accuracies on one dataset. As the curve shows, accuracy peaks in the middle ranges and dips at the two ends. In terms of generalization, the two ends denote not enough generalization and too much generalization. The optimal setting of t is closely tied to the data. It would be difficult to choose a value a priori. One possibility is to dynamically adjust t, on a motif-by-motif basis, using feedback from the classifier. This is a direction for future research.

5 Related Work

Research on moving object databases (MOD) has been an emerging field. There are several major areas in this field that relate to our work. First is nearest neighbor (NN) search [18, 5, 19, 9, 16, 7]. [18] uses a data structure similar to R-Tree to store samplings of the moving objects. [5] uses TPR-tree [15, 20] to do NN and also reverse NN. [19] uses R-Tree with a different metric to perform continuous NN queries. These works are only concerned with the proximity of moving objects in continuous settings. They are not concerned with higher level concepts of movement patterns or associations to time and location values.

There were some approaches for pattern existence queries over time series data [28, 2]. These approaches converted data into strings and used string matching algorithms to find patterns. In our approach, we use motif-oriented feature space, where each motif has its own properties which make the feature space high-dimensional.

Clustering of moving objects [8, 11] is another area in MOD. Closely related is clustering of time series, which can be viewed as a special case of one-dimensional spatio-temporal data [4]. There was also a work to cluster high-dimensional data stream [1]. These works were mainly focused on clustering the data itself, while we try to find the clusters in the meta-feature space to form a compact and generalized feature space.

There have been many works in the spatio-temporal domain which touch on related areas. Data structures such as TPR-tree [15], TPR*-tree [20], and STRIPES [14] index spatio-temporal data efficiently. Query processing and nearest neighbor queries are studied in [10, 13, 24, 22, 25, 5, 6]. These studies are focused on indexing and query processing at the level of raw points and edges. Our study is on data mining at the levels of motif-based micro-clusters.

In data mining, there have been works which focus on finding frequent patterns. [21] mines sequential patterns in spatial locations. At each location, there exists a sequence of events (*e.g.*, time, temperature recording, etc.). These events are similar to our motif attributes. The mined patterns are frequent sequences of events that occur at the locations. Another data mining problem is co-location mining. Each frequent co-location pattern is a set of frequent locations associated closely by some neighborhood function. [12, 17, 23] are works that focus on this problem. [17, 23] use Apriori-based [3] approaches to join smaller co-location patterns together. [23] performs a more complicated process by looking additionally at partial joins. [27] is a recent work that quickly finds co-location patterns by using multiway joins. In comparison with such spatial data mining studies, this study is on building up classification models by motif feature extraction and aggregation. This sets a different direction in spatiotemporal data mining and demonstrates its power and efficiency on the detection of anomaly in the moving object datasets.

6 Conclusion

In this paper, we have investigated the problem of anomaly detection in moving objects. With recent advances in surveillance technology and the imminent need for better protection of homeland security, automated solutions for detecting abnormal behavior in moving objects, such as ships, planes and vehicles, have become an important issue in intelligence and security informatics. This is a hard problem since the pattern of movement linked with the environment can become very complex. A primitive approach which analyzes the raw data will have a hard time making efficient and generalized decisions.

Instead, we propose a higher-level approach which views object paths as motif expressions, where a motif is a prototypical movement pattern. By linking the motifs with other features (both spatiotemporal and regular), we automatically extract higher level features that better represent the moving objects. In experimental testing, we see that classification using the higher level features produced better accuracies than that without using such features.

References

1. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB'04*.
2. R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. In *VLDB'95*.

3. R. Agrawal and R. Srikant. Fast algorithm for mining association rules in large databases. In *Research Report RJ 9839*, IBM Almaden Research Center, 1994.
4. A. J. Bagnall and G. J. Janacek. Clustering time series from arma models with clipped data. In *KDD'04*.
5. R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Proc. IDEAS*, 2002.
6. H. D. Chon, D. Agrawal, and A. E. Abbadi. Range and knn query processing for moving objects in grid model. In *ACM/Kluwer MONET*, 2003.
7. E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest neighbor search on moving object trajectories. In *SSTD'05*.
8. S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *KDD'99*.
9. G. S. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *VLDB'03*.
10. C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient b+-tree based indexing of moving objects. In *VLDB'04*.
11. Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD'05*.
12. K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *SSD'95*.
13. M. F. Mokbel, X. Xiong, and W. G. Aref. Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD'04*.
14. J. M. Patel, Y. Chen, and V. P. Chakka. Stripes: An efficient index for predicted trajectories. In *SIGMOD'04*.
15. S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD'00*.
16. C. Shahabi, M. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. In *GeoInformatica*, 2003.
17. S. Shekhar and Y. Huang. Discovering spatial co-location patterns: A summary of results. In *SSTD'01*.
18. Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query points. In *SSTD'01*.
19. Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB'02*.
20. Y. Tao, D. Papadias, and J. Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In *VLDB'03*.
21. I. Tsoukatos and D. Gunopulos. Efficient mining of spatiotemporal patterns. In *SSTD'01*.
22. X. Xiong, M. F. Mokbel, and W. G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE'05*.
23. J. S. Yoo and S. Shekhar. A partial join approach to mining co-location patterns. In *GIS'04*.
24. X. Yu, K. Q. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *ICDE'05*.
25. J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. Lee. Location-based spatial queries. In *SIGMOD'03*.
26. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD'96*.
27. X. Zhang, N. Mamoulis, D. W. Cheung, and Y. Shou. Fast mining of spatial collocations. In *KDD'04*.

28. Y. Zu, C. Wang, L. Gao, and X. S. Wang. Supporting movement pattern queries in user-specified scales. *IEEE Trans. Knowledge and Data Engineering*, 2003.