Translation and Rotation Invariant Mining of Frequent Trajectories: Application to Protein Unfolding Pathways

Alexander Andreopoulos[1], Bill Andreopoulos[1,2], Aijun An[1], and Xiaogang Wang[1]

[1] York University, Dept. of Computer Science, Toronto Ontario, M3J 1P3, Canada
[2] Biotechnological Centre, TU Dresden, Germany
{alekos,billa,aan}@cs.yorku.ca, stevenw@mathstat.yorku.ca

Abstract. We present a framework for mining frequent trajectories, which are translated and/or rotated with respect to one another. We then discuss a multiresolution methodology, based on the wavelet transformation, for speeding up the discovery of frequent trajectories. We present experimental results using noisy protein unfolding trajectories and synthetic datasets. Our results demonstrate the effectiveness of the proposed approaches for finding frequent trajectories. A multiresolution mining strategy provides significant mining speed improvements.

1 Introduction

There exist many situations where we are confronted with trajectories describing the movement of various objects. We are often interested in mining the *frequent trajectories* that groups of such objects go through. Trajectory datasets arise in many real world situations, such as discovering biological patterns, mobility experiments, and surveillance [9,7,4]. Of special interest are trajectories representing protein unfolding pathways, which have been derived from high-throughput single molecule force spectroscopy experiments [8]. Such trajectories are represented on a two-dimensional $force \times distance$ grid. The y axis corresponds to the force (pN) involved in pulling the protein out of the cellular membrane via the tip of a mechanical cantilever; the x axis corresponds to the force-induced distance (nm) on the unfolding pathway of the protein. Such trajectories are often very noisy, which makes it difficult to distinguish the frequent subtrajectories from the deluge of irrelevant trajectories. Moreover, frequent subtrajectories may be translated or rotated with respect to one another. Our aim is to find such frequent subtrajectories in datasets resulting from high-throughput experiments. This is useful for identifying different protein unfolding pathways and, therefore, classifying proteins based on their structure. The contributions of this paper are as follows: (*i*) We present a framework for finding frequent trajectories whose sampling interval is small enough to estimate their first and second order derivatives. (*ii*) We propose a robust framework for mining frequent translated trajectories and frequent trajectories that are both rotated and translated with respect to each other. (*iii*) We apply our method

T. Washio et al. (Eds.): PAKDD 2007, LNAI 4819, pp. 174–185, 2007.

to find frequent trajectories in protein unfolding pathways. (*iv*) We present a multiresolution framework to speed up the mining process.

This paper is organized as follows. Section 2 presents some related work. Section 3 introduces the general framework we use for mining trajectories. Section 4 describes a method for mining translated and rotated trajectories. Section 5 offers an approach for optimizing the mining speed of frequent trajectories and dealing with noisy trajectories. Section 6 presents experiments testing the proposed approaches. Section 7 concludes the paper.

2 Related Work

In sequential pattern mining we are typically given a database containing sequences of transactions and we are interested in extracting the *frequent sequences*, where a sequence is frequent if the number of times it occurs in the database satisfies a minimum support threshold. Popular methods for mining such datasets include the GSP algorithm [2] - which is an Apriori [1] based algorithm - and the PrefixSpan [10] algorithm. GSP can suffer from a high number of generated candidates and multiple database scans. Pattern growth methods such as PrefixSpan are more recent approaches for dealing with sequential pattern mining problems. They avoid the candidate generation step, and focus the search on a restricted portion of the initial database making them more efficient than GSP [2,10]. The problem that is most related to frequent trajectory mining is sometimes referred to as *frequent spatio-temporal sequential pattern mining* in the literature. The main difference between our work and previous work [9,7,4,5] is that our method assumes that we are dealing with *densely sampled* trajectories - trajectories whose sampling interval is small enough to allow us to extract from a trajectory its first and second derivative. This allows us to define a neighborhood relation between the cells making up our trajectories, allowing us to perform various optimizations. There has been a significant amount of research on defining similarity measures for detecting whether two trajectories are similar [13,3]. However, this research has not focused on mining frequent trajectories. The previous work closest to our approach is given in the innovative work described in [5] where the authors match two candidate subgraphs by comparing the set of angles made by the graph edges. This measure is similar to the curvature measure that we use later on to detect rotation and translation invariant trajectories. However, [5] is not suited for detecting trajectories that are translated but not rotated with respect to each other and does not address various robustness and speed improvements that are introduced in this paper.

3 Apriori Based Mining of Frequent Trajectories

We define a trajectory c as a continuous function $c(s) = [x(s), y(s)]$ in the 2D case and as $c(s) = [x(s), y(s), z(s)]$ in the 3D case. Similar extensions follow for higher dimensional trajectories. The function $c(s)$ is an arc-length parameterization of a curve/trajectory. In other words, the parameter s denotes the length along

the trajectory and $c(s)$ denotes the position of the trajectory after traversing distance s. In other words our trajectories do not depend on time, or the speed with which the object/person traverses the trajectory. We assume independence from time and speed for mining the frequent trajectories and subtrajectories.

A trajectory c is *frequent*, if the number of the trajectories $\{c_1, c_2, ..., c_o\}$ that pass through the path described by c satisfy a minimum support count (*minsup*). This definition requires only that there exist *minsup* subtrajectories of all trajectories in $\{c_1, c_2, ..., c_o\}$ that are identical to c; but it does *not* require that $c = c_i$ for a sufficient number of c_i's. More formally, we say that trajectory c over interval $[0, \tau]$ is frequent with respect to a dataset of trajectories if there exist a *minsup* number of compact intervals $[\alpha_1, \alpha_1+\tau], \cdots, [\alpha_{minsup}, \alpha_{minsup}+\tau]$ such that for all $i \in \{1, \cdots, minsup\}$ and for all $0 \leq s \leq \tau$ we have $c(s) = c_{\pi(i)}(\alpha_i + s)$ (where π is a permutation function of $\{1, \cdots, o\}$).

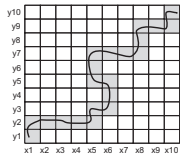

Fig. 1. A cell sequence representation of a dense trajectory. The cell sequence representation of the dense trajectory consists of the gray cells (in order) that are intersected by the dense trajectory.

The frequent trajectory mining problem for 2D trajectories can be formulated as a sequential pattern mining problem in the following way. The 3D case is similar to the 2D case. Assume that we are observing a square *region* of size $N \times N$ over which all the trajectories occur. By splitting the region into a *grid* of square *cells*, as shown in Figure 1, we denote by (x_i, y_j) the cell located at the i^{th} column and j^{th} row. A potential way of discretizing a region into a grid is by uniformly sampling along the two dimensions. In this paper we create the grid by uniform sampling, even though square cells are not necessary for our approach to work. Then we define:

(*i*) A trajectory $c(s)$ is referred to as a *dense trajectory* if it is represented by a densely sampled set of points. The sampling interval depends on the problem at hand and should be small enough to obtain accurate first and second derivatives.

(*ii*) A dense trajectory's *cell sequence* refers to the sequence of cells $((x_{\pi_x(1)}, y_{\pi_y(1)}), \cdots, (x_{\pi_x(n)}, y_{\pi_y(n)}))$ intersected by the dense trajectory (where π_x is a permutation function). The following conditions must hold: *a.* $\pi_x(i) \neq \pi_x(i+1)$ or $\pi_y(i) \neq \pi_y(i+1)$, and *b.* $|\pi_x(i) - \pi_x(i+1)| \leq 1$ and $|\pi_y(i) - \pi_y(i+1)| \leq 1$. Thus, we encode the order in which the dense trajectory intersects the cells. As we discuss below, in some situations it is preferable to also associate with each cell (x_i, y_j) from the sequence the arclength/distance over which the trajectory falls in this cell.

(*iii*) The number of cells in a trajectory's cell sequence is its *length*. For example, the cell sequence $((x_4, y_3), (x_3, y_2), (x_3, y_3), (x_3, y_4), (x_2, y_5))$ has length 5.

(*iv*) A *continuous subsequence* ω of a trajectory c's cell sequence $((x_{\pi_x(1)}, y_{\pi_y(1)}), \ldots, (x_{\pi_x(n)}, y_{\pi_y(n)}))$ must satisfy $\omega = ((x_{\pi_x(i)}, y_{\pi_y(i)}), (x_{\pi_x(i+1)}, y_{\pi_y(i+1)}), \ldots, (x_{\pi_x(j)}, y_{\pi_y(j)}))$ where $1 \leq i \leq j \leq n$.

Sometimes a trajectory $c(s)$ might be represented by a small number of sample points. We can interpolate those points and subsample the interpolated trajectory, to obtain the dense representation of those trajectories.

Using the cell representation method to represent trajectories, the problem of mining frequent trajectories is defined as finding all the contiguous subsequences of the cell sequences in a database that satisfy a support threshold. We first point out that frequent trajectories satisfy the Apriori property: Any continuous subsequence of a frequent trajectory's cell sequence is frequent. We exploit this property to implement efficient algorithms for mining frequent cell sequences.

If (x_i, y_j) is our current cell position, the next allowable cell position (x_k, y_l) *must* be one of its 8 neighboring cells, such that $|i-k| \leq 1$ and $|j-l| \leq 1$. We use this constraint to modify the GSP algorithm and generate a much lower number of candidates than the GSP algorithm would generate without this constraint.

Figure 2 shows the pseudocode for the Apriori based mining of frequent trajectories where L_k is the set of frequent length-k cell sequences found in the grid and C_k is the set of candidate length-k cell sequences. The main difference between this algorithm and GSP lies in the `trajectory()` function for generating candidates of length k from frequent cell sequences of length $k-1$. (Figure 2b). When finding the candidate length-2 cell sequences, it suffices to only join two

Inputs: *minsup*: Minimum support count.
D: Data set of cell sequences of trajectories.
Output: All the frequent cell sequences.
(1) for each cell sequence $t \in D$
(2) for each cell $g \in t$
(3) $g.count++$.
(4) $L_1 = \{cell\ g | g.count \geq minsup\}$.
//L_1 is the frequent length-1 cell sequences
//(consisting of a single cell)
(5) for (k=2;$L_{k-1} \neq \emptyset$;k++) {
(6) C_k =trajectory(L_{k-1}). //candidate //length-k cell sequences
(7) for each cell sequence representation $t \in D$ {
(8) C_t= the set of contiguous subsequences of t that are contained in C_k
(9) for each candidate cell sequence $c \in C_t$
(10) c.count++. //increment the support //count of this candidate
(11) }
(12) $L_k = \{c \in C_k | c.count \geq minsup\}$.
(13) }
(14) return $L = \cup_k L_k$.

(a)

Input: L_k: The length-k frequent cell sequences.
Output: C_{k+1}: The candidate length-$(k+1)$ cell sequences.
(1) if $(k = 1)$ {
(2) for all pairs of single cells $(a_1) \in L_1$,$(b_1) \in L_1$ such that $(a_1) \neq (b_1)$
(3) if (a_1) and (b_1) are neighbors, then
(4) $c_2 = (a_1, b_1)$. //join a_1 and b_1
(5) $C_2 = C_2 \cup \{c_2\}$.
(6) } else {
(7) for all pairs of length-k cell sequences $(a_1, \cdots, a_k) \in L_k$, $(b_1, \cdots, b_k) \in L_k$ such that $(a_1, \cdots, a_k) \neq (b_1, \cdots, b_k)$
(8) if $(a_2, \cdots, a_k) = (b_1, \cdots, b_{k-1})$, then
(9) $c_{k+1} = (a_1, \cdots, a_k, b_k)$.
(10) $C_{k+1} = C_{k+1} \cup \{c_{k+1}\}$.
(11) }
(12) return C_{k+1}.

(b)

Fig. 2. **(a)** Apriori based mining of frequent trajectories. **(b)** The `trajectory()` function.

length-1 cell sequences i.e., single cells, if the cells are neighboring/adjacent to each other, resulting in a much smaller number of candidates than if we had used GSP to accomplish this without using this neighborhood constraint. For $k \geq 2$, when joining length-k cell sequences to find length-$(k+1)$ candidate cell sequences, it suffices to only join cell sequences a with b if the last $k-1$ cells of a and first $k-1$ cells of b are identical. We notice that by joining two continuous paths, the resulting path is also continuous. Also, notice that there is no pruning step in the candidate generation process of our algorithm. This is because pruning may cause the loss of good candidates since we are mining for frequent *contiguous* subsequences. This is another major difference between the stardard GSP algorithm and this algorithm.

Assume there are r cells into which our $N \times N$ region has been split and there are b neighboring cells for each non-boundary cell. In our case $b = 8$, since each cell is surrounded by at most 8 other cells. Then, the upper bound on the number of length-$(k+1)$ candidate cell sequences generated is $|L_k| \times b$ since every sequence in L_k can only be extended by its b neighboring cells on one end of the sequence. This is lower than the upper bound of $|L_k| \times r$ that GSP might generate if we were dealing with a sequential pattern mining problem where we could not apply this neighborhood constraint, since typically $b << r$. Because of our definitions, a cell sequence of length n has $O(n^2)$ subsequences, implying that a brute force approach for finding the support count of each candidate would run in polynomial time. However, the number of database scans would be greater than the number of database scans needed by the Apriori based method.

4 Translational and Rotational Invariant Mining

In this section we present two trajectory mining techniques. The first is a method for mining frequent trajectories that are translated with respect to each other. The second method is for mining frequent trajectories that are both translated and rotated with respect to each other. Figures 3(e) and 3(f) show examples of translated and rotated trajectories, respectively. Such algorithms are useful in situations where we are interested in detecting more complex motion patterns. For example in surveillance situations, the camera which extracts the motion patterns might be rotated and translated by an unknown amount over the course of acquiring the motion patterns. In such situations the best we can hope to accomplish, in terms of frequent trajectory mining, is to make the frequent trajectory extraction invariant to the unknown amount by which the camera and subsequently the trajectories were translated and rotated.

Assume $c_1(s) = [x(s), y(s)]$ and $c_2(s) = [x(s) + 5, y(s) + 3]$. In other words c_2 is a translated version of path c_1. If we take the derivatives $c_1'(s)$, $c_2'(s)$ of these two paths then we notice that $c_1'(s) = c_2'(s)$ for all values of s. We use this fact to mine for frequent trajectories that are translated with respect to each other. An issue to keep in mind is that derivatives tend to magnify noise. In other words, two trajectories that are slightly different due to noise would have an even more different derivative. Below we will discuss methods for dealing with

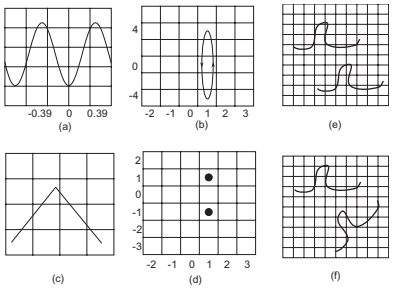

Fig. 3. (*a*) A trajectory given by function $c(s) = (s, sin(4s))$. (*b*) The derivative space $c'(s) = (1, 4cos(4s))$. We mine for such patterns to find translated patterns. (*c*) A trajectory that is not differentiable everywhere. (*d*) The derivative space of this function consists of two isolated and non-neighboring cells. We either have to smooth the function in (*c*) to have a continuous derivative space, or we are forced to use standard Apriori to do the mining, which is much slower. (*e*) Trajectories that are translated with respect to each other. (*f*) Trajectories that are both translated and rotated.

this problem. We mine for frequent translated trajectories in the following way. For every dense trajectory $c_i(s) = [x_i(s), y_i(s)]$ in our database of trajectories we use finite differences to find its derivative $c_i'(s) = [x_i'(s), y_i'(s)]$. We then use equiwidth binning to discretize the space of derivatives for the x-coordinates and y-coordinates into a number of bins (cells). We then represent every $c_i'(s)$ by a sequence of tuples (dx_i, dy_i), each of which denotes the current *derivative cell* in which the trajectory is located. A new tuple (dx_i', dy_i') is added to the sequence of tuples whenever the trajectory's derivative changes significantly enough to be part of a new derivative cell (Figure 3). For example, a linear trajectory is encoded by a sequence of length 1 (a single derivative cell) since its slope is constant. With each such tuple we could also associate a number denoting the arclength/distance over which the cell occurs. We use this measure, as described below, to detect translated trajectories. We can apply on this new trajectory representation the trajectory mining algorithms to find frequent trajectories that are translated with respect to each other.

Note that in our test cases we make the assumption that we are dealing with differentiable functions that do not change abruptly. Figures 3(c) and 3(d) show problems that might arise otherwise. If we wish to mine such trajectories, we could either apply some sort of smoothing such as the wavelet transformation described in the next section to make the function better behaved, or we could apply some sort of standard Apriori/GSP/PrefixSpan mining which does not make the neighborhood assumption for adjacent cells in our cell sequence representation. This would likely be detrimental to our trajectory mining speed.

The *curvature* of an arclength parameterized path $(f(s), g(s))$ at s is given by the derivative with respect to s of the angle θ the path makes with the x axis.

$$\kappa = \frac{d\theta}{ds} = \begin{vmatrix} f' & g' \\ f'' & g'' \end{vmatrix} = f'g'' - f''g' \tag{1}$$

Intuitively, the curvature gives us a measure of the rate with which a curve is changing direction. It is straightforward to show that any rotation and translation of $(f(s), g(s))$ results in the same curvature measure. In other words *curvature for 2D trajectories is rotation and translation invariant.* We can, therefore, use this measure to detect rotationally and translationally invariant patterns in a similar way as we did with translated trajectories. To encode each trajectory $c_i(s)$ we follow the same procedure that we followed for the translationally invariant mining. We can use equiwidth or equidepth binning to discretize the space of curvature measures and encode each trajectory using a sequence of *curvature cells*. We could also associate a number with each curvature cell, denoting the arclength/distance over which the trajectory belongs in this cell before it changes significantly to warrant using another cell in the sequence to encode it.

We wish to point out a potential problem which we have not discussed so far. The problem arises in the above cell sequence representation of derivatives and curvatures if we do not associate the distance over which each dense trajectory belongs to a particular cell. It is possible, for example, to have two trajectories whose cell derivative representation consists of the same two cells. If the distance over which each dense trajectory belongs to each cell is very different, the two trajectories might be very different and should not lead to a match. A solution to this problem is to associate with each cell the distance over which the dense trajectory belongs to the cell and mine this data as an extra dimension in our trajectory, or to simply use a brute force approach to refine the mined frequent trajectories. From our experiments we notice that a brute force approach is feasible in most cases due to the significantly decreased number of trajectories that need to be processed after the initial mining.

It should also be pointed out that because we are dealing with arclength parameterized curves $c(s) = [x(s), y(s)]$ we have $\int_0^s \sqrt{(x'(t))^2 + (y'(t))^2}dt = s$ which implies $\sqrt{(x'(s))^2 + (y'(s))^2} = 1$ which in turn implies $|x'(s)| \leq 1$ and $|y'(s)| \leq 1$. In other words, arclength parameterized curves do not change abruptly, implying that this parameterization makes it feasible for us to discretize the space of derivatives, since all derivative values will be in the range of -1 to 1. If we did not have such a bound on the space of derivatives this approach would be problematic in our opinion, since it would be too difficult to appropriately discretize the real line using a finite number of cells.

5 Wavelet Based Optimization of Mining Speed

Multiresolution techniques are well known in the signal processing community and are of great use for solving difficult problems such as image denoising and image compression[6]. More recently, the applicability of such methods has been demonstrated for various data mining problems. For example WaveCluster is a multiresolution clustering algorithm that uses the Wavelet transform to transform the original data and find dense regions in the transformed space [12]. We now propose a method for speeding up the frequent trajectory mining phase by mining for trajectories on multiple resolutions. Assume **dwt** is a function

denoting the 1-D discrete wavelet transform. For example, given as input a vector x, **dwt**(x) returns a vector x' denoting the lower resolution version of vector x. At the pre-processing stage when the original *dense* trajectories are processed in order to convert them to their cell sequence representation, we apply **dwt** to each dense trajectory in order to get its lower resolution version. Then, we convert the lower resolution trajectory to a lower resolution cell sequence representation, where the cells now have *twice the width and height* they previously had. The effect of this is that the lower resolution cell sequence representation has approximately *half the length* of the original cell sequence representation. We refer to these new cell sequence representations as *scaled down* and we refer to their larger cells as *scaled down cells*. We can use these scaled down cell sequence representations to mine the frequent trajectories in our database at a coarser scale. This significantly decreases the length of our trajectories and the number of cells used to represent our region. As we discuss below, this can lead to significant improvements in the mining speed. A potential objection to this procedure involves the need to apply the wavelet transform. Some might argue, that in order to get a smaller cell representation, it suffices to simply double the cell size used in our equiwidth/equidepth discretization of the trajectories. The reason is to remove noise and high frequency components. This makes the derivative estimates more accurate. It also diminishes the risk that for various signals whose dense representation has a localized high frequency component passing near a cell border, we would needlessly add cells to a trajectory's cell sequence representation. Furthermore, by 'smoothing' a function we decrease the risk of dealing with functions which are not differentiable everywhere (Figures 3(c), 3(d)). A drawback of this method is that we lose precision on the localization of the trajectory since the scaled down cell sequence representations consist of larger cells. We now propose a method for obtaining a better localization of the coordinates through which frequent trajectories pass. When mining for frequent trajectories we are often only interested in finding trajectories that have a minimum non-scaled down length of m. Let S denote the set of all scaled down cells through which a frequent scaled down trajectory of length at least $\lfloor \frac{m}{2} \rfloor$ passes. Then, we are guaranteed that the frequent trajectories with a non-scaled down cell sequence representation of length over m pass through the scaled down cells in S. This means that to refine the accuracy of mining frequent scaled down cell sequence representations, it suffices to consider only the non-scaled down cell sequence representations that pass through some cells in S. For datasets that have very spread out trajectories with a few paths through which frequent trajectories pass, this can also result in significant improvements in mining speed.

6 Experiments

We experimentally evaluated the above mentioned algorithms using a real world dataset containing 139 protein unfolding trajectories, acquired using single molecule force spectroscopy [8]. We also use a dataset of 4,000 synthetic trajectories. In both cases, we are interested in mining translation invariant

trajectories. We generated the synthetic trajectories with the same trajectory simulator that was used by the inventors of the TPR-tree algorithm [11] for indexing moving objects. We used MATLAB 7.01 running on an Intel Xeon 3Ghz with 3GB RAM to run our experiments. We utilized progressively larger subsets of the above mentioned trajectories to investigate the accuracy of our algorithms, their scaling properties, and their robustness to noise.

The distinguishing characteristic of the protein unfolding dataset was its extremely noisy trajectories. Moreover, in this dataset it is desirable to find subtrajectories that are translation invariant. The dataset consisted of 139 trajectories which we denoised, before applying the translation invariant methodology to detect the frequent translation invariant subtrajectories. We discretized the derivative space into 8x8 cells, and each trajectory's derivative representation consisted of around 200 cells on average. This served to demonstrate the practicality of our approach on a real world problem and to obtain quantitative results of our translation invariant method's performance on a very noisy dataset.

Fig. 4(a) shows a noisy protein unfolding trajectory. Fig. 4(b) shows the denoised trajectory from (a) using wavelet analysis. As shown, wavelet analysis was successful in identifying peaks, associated with single potential barriers stabilising segments within membrane proteins. Figures 4(c)-(f) show matched subtrajectories between the red arrows. Matched subtrajectories may potentially be manually annotated peaks, corresponding to three-dimensional protein structures [8]. On close inspection of the results we notice that the algorithm is

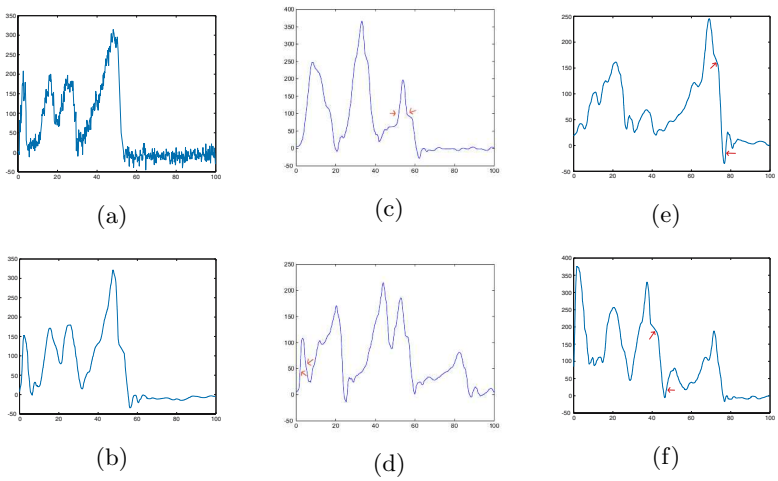

Fig. 4. (a):Noisy protein unfolding trajectory. (b):Denoised trajectory. Trajectories where a translation invariant match was found, with support threshold 3% : (c,d): the twin peaks with x-coordinates approximately in intervals [52, 55] in (c) and [2, 4.7] in (d) (interval denoted with red arrows), are matched as frequent translated subtrajectories. (e,f): In (e) x-coordinate interval [74, 78] matches in (f) interval [43, 47]. The y axes correspond to the force (pN) and the x axes correspond to the force-induced distance (nm) on the unfolding pathway of the protein.

Table 1. Results using the 139 protein unfolding trajectories (translation invariant with standard apriori/not translation invariant with standard apriori)

Number of Trajectories		Support %		
		1	2	3
D=139	T	22693/16054	7411/7579	3206/3792
	L	39/61	28/43	25/35
	F	45677/44937	15537/21507	6617/10105

Table 2. Results using the translation invariant/rotation invariant mining on a synthetic dataset

Number of Trajectories		Support %		
		1	2	3
D=1000	T	33351/13	12843/11	6565/11
	L	43/21	32/21	25/21
	F	35117/338	11267/312	3205/288
D =2000	T	53071/24	24984/22	12026/18
	L	33/21	30/21	12/19
	F	27007/338	10705/312	2515/200
D=4000	T	110824/44	47301/41	23934/35
	L	32/21	30/21	11/19
	F	20128/312	9613/288	1832/200

successful at detecting translated subtrajectories. Table 1 presents some quantitative results with and without the translation invariant mining algorithm. It shows the running time in seconds (T), the length of the longest frequent trajectory discovered (L) and the total number of frequent trajectories discovered (F). The "Support" columns show the minimum support used, as a percentage of the number of trajectories (D).

The next set of experiments was designed to test the translation and rotation invariant mining of trajectories (Section 4). For the translated trajectories, the derivative values of the trajectories were discretized into a 32×32 grid using equidepth bining and the curvature values were split into 32 bins using equiwidth bining. We first applied our mining method to mine these derivative cell representations. See Table 2 for the results using the translation invariant algorithm. The algorithm accurately detected the frequent translated trajectories. In the next section we show how the mining speed could be improved. To test the rotation invariant algorithm, we added to each of the above datasets 100 rotated sinusoidal trajectories having various frequency values in order to obtain a better understanding of how the algorithm performs for longer trajectories. The rotation invariant algorithm is much faster than the other approaches. This is due to the fact that the curvature at any point along a 2D curve is described by a single number while the translation invariant approach and Standard Apriori mining approach using 2D trajectories required two real numbers to describe each point along the curve. Furthermore, on average the curvature cell trajectories were

Table 3. Results using the wavelet based approach on a synthetic dataset. Shows the running time with/without the optimization.

Number of Trajectories		Support %		
		1	2	3
D=1000	T	224/630	131/344	95/240
	L	20/20	20/20	20/20
	F	1057/1133	742/638	634/534
D =2000	T	436/1270	262/671	192/474
	L	20/20	20/20	20/20
	F	1008/1088	738/619	631/526
D=4000	T	887/2563	542/1327	388/951
	L	20/20	20/20	20/20
	F	998/1099	743/614	630/532

much shorter in length resulting in faster mining speeds. We also observe that curvature is very sensitive to noisy trajectories, as expected. This indicates that a preprocessing stage for denoising/smoothing the trajectories is a necessity.

We then proceeded to test the wavelet based method for optimizing the mining speed (Section 4). Table 3 presents our results. The results demonstrate that the wavelet based method significantly improves the mining speed if there are clusters through which frequent trajectories of the desired length m pass, since it provides a quick method of finding which trajectories should be eliminated from further processing. However, if our trajectories are evenly spread out around the region we do not expect to gain a lot in terms of mining speed in general.

To test the approach we used the Standard Apriori based algorithm to compare the mining performance with and without the wavelet based method described in Section 4. In order to demonstrate the effectiveness of the approach we slightly modified the datasets we had used to test the Standard Apriori algorithm. We translated one-fourth of the trajectories in each of the datasets so that they are close to each other and form a cluster of trajectories that pass close by each other and added to this cluster a sufficient number of duplicate trajectories of length at least 15, so that they would end up being discovered as frequent trajectories during the mining process. Note that these were also the longest trajectories in our dataset. The other three-fourths of the trajectories were positioned in locations away from this cluster and were spread out so that it was unlikely to have many frequent trajectories in this group. We instructed our algorithm to search for frequent trajectories of length at least 15. We used a grid spacing of 32×32 cells to represent the scaled down cells and used a 64×64 grid to represent the non-scaled down cells. The algorithm located the trajectories of length 15 while significantly improving the mining speed.

7 Conclusions

We presented various methods for mining frequent trajectories that are translated and/or rotated with respect to each other. We also presented approaches

for optimizing the mining speed of such trajectories. The methods were successful in finding peaks in protein unfolding pathways, which may correspond to three-dimensional protein structures. More research needs to be done in using intelligent methods for discretizing the continuous range of values that the trajectories can assume, as this could potentially decrease the number of cells used to encode trajectories.

Acknowledgements

The authors are grateful for the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through the PGS-D program and the Ontario Graduate Scholarships program (OGS). We would also like to thank Annalisa Marsico for kindly providing the protein unfolding dataset.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. Proc. 20th Int. Conf. Very Large Data Bases (1994)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE (1995)
3. Cai, Y., Ng, R.: Indexing spatio temporal trajectories with chebyshev polynomials. In: SIGMOD (2004)
4. Cao, H., Mamoulis, N., Cheung, D.: Mining frequent spatio-temporal sequential patterns. In: Proceedings of the ICDM (2005)
5. Kuramochi, M., Karypis, G.: Discovering frequent geometric subgraphs. In: 2nd IEEE Conference on Data Mining (ICDM) (2002)
6. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press, London (1999)
7. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.L.: Mining, indexing, and querying historical spatiotemporal data. In: Conference on Knowledge Discovery in Data (2004)
8. Marsico, A., Labudde, D., Sapra, T., Muller, D.J., Schroeder, M.: A novel pattern recognition algorithm to classify membrane protein unfolding pathways with high-throughput single molecule force spectroscopy. Bioinformatics (2006)
9. Morimoto, Y.: Mining frequent neighboring class sets in spatial databases. In: Conference on Knowledge Discovery in Data (2001)
10. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. Int. Conf. on Data Engineering (ICDE) (2001)
11. Saltenis, S., Jensen, C., Leutenegger, S., Lopez, M.: Indexing the positions of continuously moving objects. In: SIGMOD, pp. 331–342 (2000)
12. Sheikholeslami, G., Chatterjee, S., Zhang, A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. In: Proceedings of the 24th International Conference on Very Large Databases (1998)
13. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.: Indexing multidimensional time-series with support for multiple distance measures. In: SIGKDD 2003 (2003)