Classification and Prediction of Motion Trajectories using Spatiotemporal Approximations

Andrew Naftel & Shehzad Khalid
School of Informatics
University of Manchester
Manchester, M60 1QD
a.naftel@manchester.ac.uk; s.khalid-2@postgrad.manchester.ac.uk

Abstract

A new technique is proposed for classification and similarity retrieval of video motion clips based on spatio-temporal object trajectories. The trajectories are treated as motion time series and modelled using orthogonal basis polynomial approximations. Trajectory clustering is then carried out to discover patterns of similar object motion behaviour. The coefficients of the basis functions are used as input feature vectors to a Self-Organising Map which can learn similarities between object trajectories in an unsupervised manner. Clustering in the basis coefficient space leads to efficiency gains over existing approaches that encode trajectories as discrete point-based flow vectors. Experiments on pedestrian motion data demonstrate the effectiveness of our approach leading to accuracy improvements in trajectory prediction. Applications to motion data mining in video surveillance databases are envisaged.

1 Introduction

An increasing number of systems are now able to capture and store data about object motion such as those of humans and vehicles. This has acted as a spur to the development of sophisticated content-based visual data management techniques. General purpose tools are now urgently required for motion data search and retrieval, discovery and grouping of similar motion patterns, detection of anomalous behaviour, motion understanding and prediction.

Much of the recent research focus has been on representation schemes for motion indexing and retrieval [1]-[11]. This work presupposes the existence of some low-level tracking scheme for reliably extracting object-based motion trajectories. A description of relevant tracking algorithms is not within the scope of this paper but recent surveys can be found in [12, 13]. The literature on trajectory-based motion understanding and pattern discovery is less mature but advances using learning Vector Quantization (LVQ) [14], Self-Organising Feature Maps (SOMs) [15, 16], hidden Markov Models (HMMs) [17], and fuzzy neural networks [18] have been reported. Most of these techniques attempt to learn high-level motion behaviour patterns from sample trajectories using discrete point-based flow vectors as input to the learning phase. For realistic motion sequences, convergence of these techniques is slow and the learning phase is usually carried out offline. This is due to the high dimensionality of the input feature space. In this paper, we introduce a dimensionality reduction technique for learning trajectory patterns based on a spatio-temporal modelling scheme previously used for time series indexing. The resulting efficiency gains make this approach feasible to implement online.

Related work within the temporal database community on approximation schemes for indexing time series data is highly relevant to the parameterisation of object trajectories. However, computer vision researchers have been slow to appreciate the relevance of this work. For example, spatio-temporal trajectories have been successfully modelled using discrete Fourier transforms (DFT) [19],

discrete wavelet transforms (DWT) [20], adaptive piecewise constant approximations (APCA) [21], and Chebyshev polynomials [22], to name but a few.

In this paper, we aim to apply time series indexing of spatiotemporal trajectories to the problem of trajectory classification and show how to learn motion patterns by using the indexing scheme as an input feature vector to a neural network learning algorithm. The remainder of the paper is organized as follow. We review some relevant background material in section 2. In section 3, we present our trajectory modelling approach. The algorithm for learning trajectories is then presented in section 4 within the framework of a self-organising map. This is applied in the context of clustering motion trajectories and experimental results for a pedestrian object tracking database are reported in section 5. The paper concludes with a discussion of the advantages of our proposed technique over competing approaches and outlines further work.

2 Review of Previous Work

Motion trajectory descriptors are known to be useful candidates for video indexing and retrieval schemes. Previous work has sought to represent moving object trajectories through piecewise linear or quadratic interpolation functions [1, 2], motion histograms [4] or discretised direction-based schemes [3, 8, 9]. Spatiotemporal representations using piecewise-defined polynomials were proposed by Hsu [6], although consistency in applying a trajectory-splitting scheme across query and searched trajectories can be problematic. Affine and more general spatiotemporally invariant schemes for trajectory retrieval have also been presented [5, 7, 10]. The importance of selecting the most appropriate trajectory model and similarity search metric has received relatively scant attention [11].

In addition to polynomial models, a wide variety of basis functions have been used to approximate object trajectories [19, 20, 22]. Efficient indexing schemes can then be constructed in the coefficient space of the basis functions. These schemes have been compared with respect to search pruning power, CPU and I/O efficiency costs [21, 22].

It is surprising to find that many of these candidate time series indexing schemes have not yet been applied to the problem of motion data mining and trajectory clustering. Recent work has either used probabilistic models such as HMMs [17] or discrete point-based trajectory flow vectors [14, 16, 18] as a means of learning patterns of motion activity. Point-based flow vectors generally consist of spatial coordinates augmented by instantaneous object velocities and accelerations. These can be normalised to account for variations in trajectory length.

The contribution of this paper is to show that a trajectory-encoding scheme based on input feature vectors consisting of basis function approximation coefficients can be used to learn motion patterns more effectively. Hence, clustering and classification processes can be carried out more efficiently in the basis coefficient space.

3 Trajectory Representation

3.1 Least Squares Polynomials

The output of a motion tracking algorithm is usually a set of noisy 2-D tracker points (x_i, y_i) representing the object's motion path over a sequence of n frames, where $i = 1,\ldots,n$. Often the representative point is taken to be the centroid or edge midpoint of the object's minimum bounding rectangle. The motion trajectory can be considered as two separate 1-dimensional time series, $<t_i, x_i>$ and $<t_i, y_i>$, the horizontal and vertical displacement against time where $t_1 < \ldots < t_n$. We consider three alternative trajectory models: Least Squares (LS), Chebyshev polynomial approximations (CS) and discrete Fourier transform (FS). LS polynomials are suitable for modelling simple motion trails in the spatial domain, e.g. vehicles moving uniformly along highways, or for smoothing x-y projections of more complex spatio-temporal trajectories. Chebyshev approximations are more appropriate for modelling highly complex spatiotemporal trajectories such as pedestrian motion exhibiting stop-start

and looping motions, whilst Fourier series approximation are suitable for mixed types of trajectory. We compare the performance of the 3 different modelling approaches in section 5.

The trajectory can be approximated by a polynomial $P_m(t)$ of degree $m < n$ as

$$[x \mid y] \approx P_m(t) = a_0 + a_1 t + \ldots + a_m t^m \quad \textbf{(1)}$$

The projections in (x, t) and (y, t) planes are modelled as independent polynomials $P_m{}^x$, $P_m{}^y$ in t. Note that separate 1-D trajectories are created for each spatial coordinate [x | y]. The unknown 2(m+1) coefficients $\{a_{xi}, a_{yi}\}$, i = 0,...,m can be determined using LS by minimising the function E with respect to a_0, a_1, ...

$$E(a_0, a_1, \ldots, a_m) = \sum_{i=1}^{n} \{[x_i \mid y_i] - (a_0 + a_1 t + \ldots + a_m t_i{}^m)\}^2 \quad \textbf{(2)}$$

The motion trajectories are thus indexed by vector of coefficients $\{(a_{x0},\ldots,a_{xm}),(a_{y0},\ldots,a_{ym})\}$.

3.2 Chebyshev Polynomials

Alternatively, a spatiotemporal trajectory can be approximated by a function $f(t)$ expressed as a weighted sum of Chebyshev polynomials $C_k(t)$ up to degree m, defined as

$$[x \mid y] = f(t) \approx \sum_{k=0}^{m} b_k C_k \quad \textbf{(3)}$$

where $C_k(t) = \cos(k \cos^{-1}(t))$ and

$$b_0 = \frac{1}{m} \sum_{k=1}^{m} f(t_k),\ b_i = \frac{2}{m} \sum_{k=1}^{m} f(t_k) C_i(t_k) \quad \textbf{(4)}$$

for $t \in$ [-1,1] and i = 1,..., m. The k roots of $C_k(t)$ are given by t_j for $1 \le j \le k$. Implementation details can be found in [23]. Occasionally, it may be possible to approximate the motion trail (spatial trajectory shape only) in the x-y plane. In this case, we would replace t by x or y in one of the above equations depending on the choice of principal axis [6]. This would only be worthwhile if all trajectories could be aligned with the same principal axis. An example would be the modelling of vehicle trajectories in highway traffic surveillance.

3.3 Discrete Fourier Transform (DFT)

The DFT coefficients of the spatiotemporal trajectory can be calculated using the well known Fast Fourier Transform (FFT). The formulas for evaluating the Fourier coefficients can be found in [19, 23].

3.4 Similarity Search Metric

A Euclidean distance is used as the basis for comparing the similarity of motion trajectories. Each approximation produces a vector of coefficients which can be used to index a 2-dimensional spatiotemporal trajectory. Given two trajectories Q and S, we can index these by a vector of 2(m+1) coefficients $Q = \{\vec{q_0},\ldots,\vec{q_m}\}$ and $S = \{\vec{s_0},\ldots,\vec{s_m}\}$, where $\vec{q_i}$, $\vec{s_i}$ are $\vec{q_i} = [q_{xi}, q_{yi}]^T$ and $\vec{s_i} = [s_{xi}, s_{yi}]^T$ (i = 0,..., m).

A Euclidean distance function (ED) on the coefficient space can be expressed as

$$ED(Q,S) = \sqrt{\sum_{i=0}^{m} (\vec{q}_i - \vec{s}_i)^2} = \sqrt{\sum_{i=0}^{m} (q_{xi} - s_{xi})^2 + (q_{yi} - s_{yi})^2} \quad (5)$$

4 Learning Trajectory Patterns Using Self-Organizing Maps

Self-organised maps (SOMs) have been previously used for motion trajectory clustering and classification [15, 16] with trajectories encoded as discrete point-based flow vectors. This step can be replaced by the proposed coefficient indexing scheme. A SOM can discover the underlying structure of motion trajectory in an appropriate feature space through unsupervised learning. As we shall see the feature space can be the original high dimensional trajectory space or the reduced coefficient subspace.

4.1 Network Model

The architecture chosen for the SOM is very simple with a layer of input neurons connected directly to a single 1-D output layer. Each input neuron is connected to every output neuron with the connection represented by a weight vector. The network topology is shown in Fig. 1. A similar architecture was used in [16] for learning vehicle trajectories as a means for accident prediction.

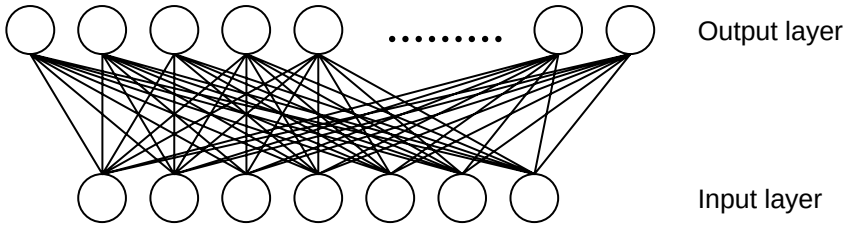


Fig. 1. SOM network architecture used for trajectory clustering

In a SOM network, physically adjacent output nodes encode the patterns in the trajectory data that are similar and, hence, it is known as a topology-preserving map. Consequently, similar object trajectories are mapped to the same output neuron. The number of input neurons is determined by the size of the feature vector which in this case relates to the selected number of coefficients for the basis functions. The degree of the polynomial can be chosen by setting a threshold on the maximum deviation of the function approximation from the data or mean-squared error. The number of output neurons represents the number of clusters in the trajectory data and this is selected manually.

4.2 Learning Algorithm

The algorithm used to cluster the trajectories differs slightly from the original SOM proposed by Kohonen [24]. The number of output neurons is initially set to a higher value than the desired number of clusters which we wish to produce. After training the network, clusters representing the most similar patterns are merged using an agglomerative method until the cluster count is reduced to the required number. The weights are initialised to linearly spaced values lying within the range of input values. Neighbourhood size is initially set to cover over half the diameter of the output neurons.

Let B be the input feature vector representing the set of trajectory basis function coefficients, and W the weight vector associated to each output neuron. The learning algorithm comprises the following steps:

1. Determine the winning output node k (indexed by c) such that the Euclidean distance between the current input vector B and the weight vector W_k is a minimum amongst all output neurons, given by the condition

$$\| B - W_c(t) \| \leq \| B - W_k(t) \| \qquad \forall k \tag{6}$$

2. Train the network by updating the weights. A subset of the weights constituting a neighbourhood centred around node c are updated using

$$W_k(t+1) = W_k(t) + \alpha(t)\eta(k,c)(B - W_k(t)) \tag{7}$$

where $\eta(k, c) = \exp(-|r_k - r_c|^2 / 2\sigma^2)$ is a neighbourhood function that has value 1 when $k=c$ and falls off with distance $|r_k - r_c|$ between nodes k and c in the output layer, σ is a width parameter that is gradually decreased over time and t is the training cycle index.

3. Decrease the learning rate $\alpha(t)$ linearly over time.

4. After a pre-determined number of training cycles, decrease the neighbourhood size.

5. At the end of the training phase, merge the most similar cluster pairs until the desired number of groupings is achieved. Clusters are merged by calculating the weighted mean of the weights associated with each neuron taking into account the number of input samples allocated to the cluster. Assuming W_a and W_b are the weight vectors associated with output neurons representing the most similar clusters, and m, n are the number of sample trajectories mapped to these neurons respectively, a new weight value W_{ab} for the merged cluster can be calculated as

$$W_{ab} = \frac{mW_a + nW_b}{m+n} \tag{8}$$

5 Experiments

We now present some results to indicate the effectiveness of the proposed trajectory clustering technique. We have evaluated the performance of the trajectory clustering algorithm using the CAVIAR visual tracking database [26]. The database consisted of hand annotated video sequences of moving and stationary people and are intended to provide a testbed for benchmarking vision understanding algorithms. Semantic descriptions of the target object behaviours and motion had been previously generated and stored in XML files. These files have been parsed to extract ground truth-labelled object trajectories. The dataset contains 222 trajectories as shown in Fig. 2.

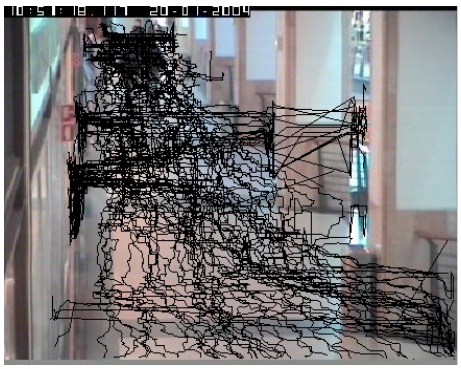


Fig. 2. Background scene containing ground truth labelled object trajectories extracted from the CAVIAR dataset [26].

The performance of the 3 different trajectory representation schemes have been compared. The purpose of the experiment is to test the retrieval accuracy for each approximation scheme and to

investigate the effect of varying the number of coefficients used for model fitting. We also wish to examine the effect on retrieval performance by introducing additive noise. A corrupted dataset S_C is produced by adding the term η*U[0,1]**rangeValues* to each coordinate in the original set S, where η is a scaling factor such that $0 \leq \eta \leq 1$, U[0,1] is uniform random noise on the interval [0,1], and *rangeValues* is the range on X or Y coordinates.

Each corrupted trajectory in S_C then serves as an example query Q_C and we search for its closest match in the original dataset S by searching for minimum $ED(Q_C,S)$. A set of rankings over all Q is produced. In the absence of noise and when no data is excluded, the mean ranked value should be 1. For ease of comparison we record the proportion of times (as a percentage) the query trajectory is ranked correctly as 1. This is repeated for different number of coefficients in LS, Chebyshev and Fourier approximations and for various values of η. The results are summarised in Fig. 3.

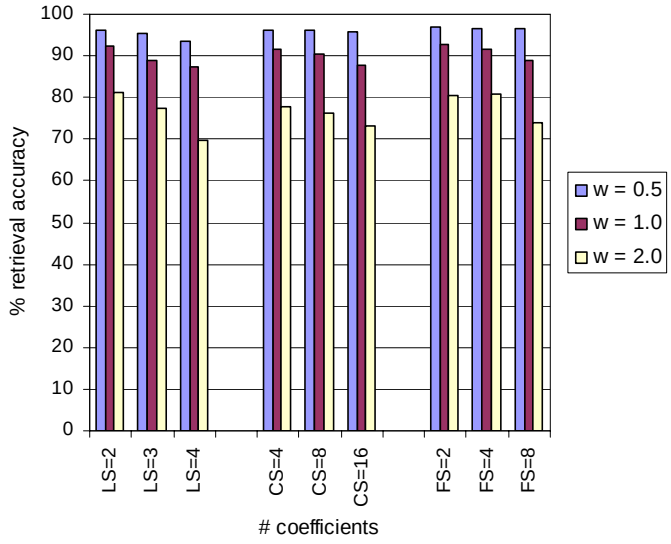


Fig. 3. Effect of scaled uniform noise on trajectory retrieval accuracy. *w* is the scaling factor, LS= least squares, CS = Chebyshev polynomials, and FS = Fourier series approximation.

For small amounts of noise, the choice of approximation scheme or number of coefficients does not appear to be too critical, although there is a slight fall off in performance of LS as the coefficient number increases. For higher noise levels, it is apparent that Fourier series approximations outperform LS and Chebyshev polynomials. This may be explained by the fact that Euclidean distance defined over Fourier coefficients is more noise resistant in the frequency domain. In previous work, it has been shown that a LS-RANSAC approach would be beneficial if it is known that the tracking algorithm produces very noisy estimates with a significant number of outliers [11].

The retrieval experiment was then repeated but this time under simulated object occlusion by removing at random a subsequence of points. The proportion of points removed (p) varied between 10, 20 and 30% of the trajectories' length. Each of the results obtained were averaged over 10 random occluding subsequences. In this instance there was no added noise. The percentage retrieval accuracy over all query trajectories was determined as before for each choice of approximation scheme and number of coefficients. The results are shown in Fig. 4. In this case LS approximations perform best under conditions of simulated occlusion.

The *ED*-metric defined over the coefficient subspace is now used to perform the trajectory clustering step. We ran the SOM clustering algorithm using the 3 different spatiotemporal approximations. From empirical observations it was noticed that if the number of coefficients is too low (typically $m < 3$), poor clustering results are obtained. As a sanity check, we repeated the clustering using a standard K-Means algorithm [25] and the same result was observed. Although satisfactory results are obtained in retrieval experiments with a small number of coefficients, there is insufficient discriminatory power in a very low dimensional coefficient subspace to achieve a meaningful clustering outcome.

In practice we have found no discernible differences in SOM clustering results between spatiotemporal trajectory models generated by CS=4..8 and FS=4..6 for about K=10 clusters. The

SOM algorithm always produces visually better cluster separations than *K*-means. This is to be expected given that SOM better preserves the topology of the original trajectory space. We do not attempt to normalise to achieve scale or translational invariance since we wish to preserve these differences in the clustering step.

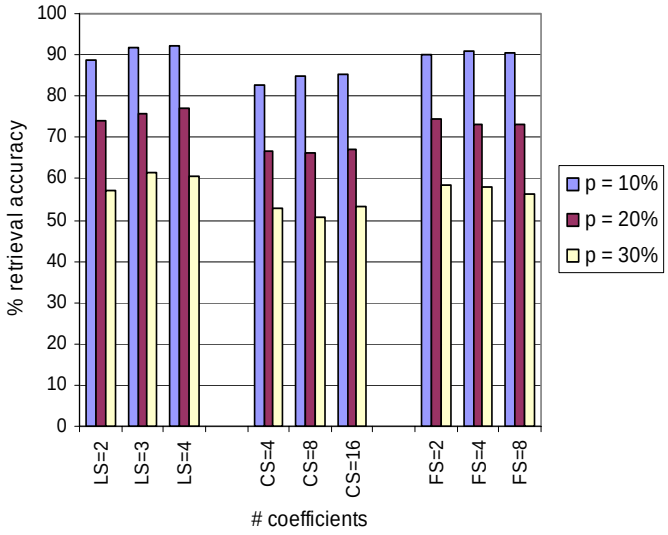


Fig. 4. Effect of occluding subsequences on trajectory retrieval accuracy. *p* is the percentage of subsequence length removed from the original trajectory. Results are averaged over 10 random removals. LS= least squares, CS = Chebyshev polynomials, and FS = Fourier series approximation.

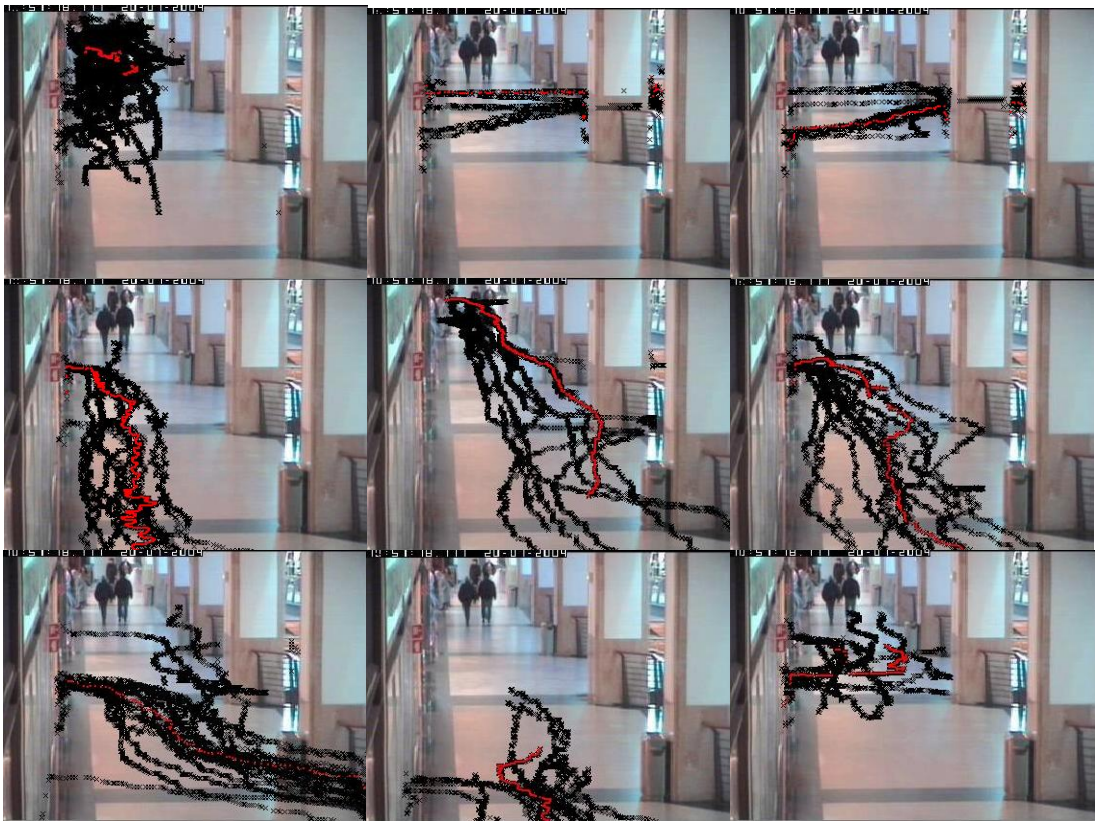


Fig. 5. Clustering of spatiotemporal object trajectories from CAVIAR database using SOM algorithm and Chebyshev polynomial of degree 8. The trajectory closest to the codebook vector is displayed in red**.**

We initially train a SOM network with 50 output neurons and then reduce these to 9 using the agglomerative clustering method described in section 4.2. The resulting trajectory groups are shown in Fig. 5 for Chebyshev polynomial of degree 8, although similar results are obtained using Fourier approximations of degrees 4 to 6. Visual inspection shows that qualitatively similar motion trajectories have been grouped together quite successfully. Motions across the corridor from left-to-right and right-to-left are grouped into separate clusters as expected.

In order to visualise the effects of trajectory clustering in the coefficient subspace, we have performed Principal Component Analysis (PCA) on the vector of Fourier coefficients (m = 4) and calculated the first 3 PCs which explain 94% of the total variation. The results are shown in Fig. 6. Each point represents an instance trajectory and these are colour/marker coded to highlight the separate cluster groups each trajectory is allocated to. These plots show a good degree of cluster separation in the PCA reduced coefficient subspace.

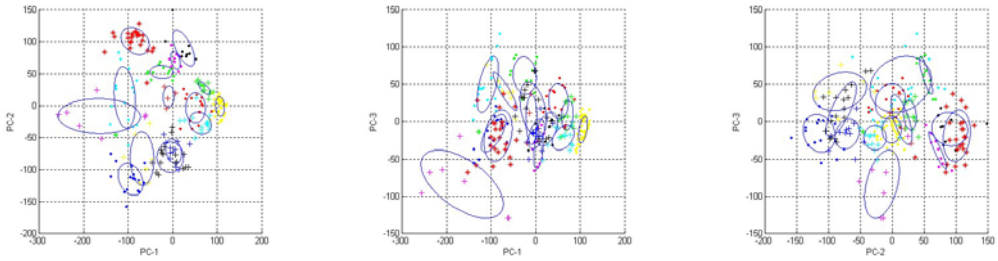


Fig. 6. Trajectory clustering in the Fourier coefficient subspace (m = 4). For visualisation purposes we have performed PCA on the coefficient vector and plotted PC_1 vs PC_2 (left), PC_1 vs PC_3 (centre), and PC_2 vs PC_3 (right). Error ellipses for the covariance matrix are shown for each cluster group.

To investigate the effectiveness of clustering in the dimensionally reduced coefficient subspace compared to clustering in the original space using trajectories encoded as discrete point-based flow (PBF) vectors, we performed some classification tests. The class labels of the motion trajectory patterns were learned using the SOM and *K*-means unsupervised techniques on the CAVIAR training data. The dataset Ξ was then randomly partitioned into training and test sets of equal sizes for cross-validation purposes. We used a *k*-NN classifier (with k = 1) to classify instance trajectories from the test set and generated the overall classification accuracy. To avoid bias, we repeated the random partitoning 500 times and averaged the classification errors over the test set. The results summarised in Table 1 demonstrate the superiority of learning trajectory patterns in the coefficient subspace. The classification accuracy obtained using coefficient subspace learning is higher than that of discrete PBF vector encoding for both SOM and *K*-means algorithms.

Method type	Accuracy %
SOM: coefficient subspace	91.9
SOM: PBF vectors	79.9
K-Means: coefficient subspace	88.8
K-Means: PBF vectors	85.6

Table 1. Comparison of mean overall classification accuracy for 2 different clustering techniques (SOM and *K*-means) and 2 different trajectory encodings (coefficient subspace and discrete PBF vectors). #classes : #trajectories = 9 : 111.

Significant speed-ups are thus possible using this approach and it now becomes feasible to learn motion activity patterns and perform trajectory classification online. This is not currently possible when trajectories are encoded as discrete PBF vectors.

In the next experiment we compare the performance of all 4 methods in trajectory prediction and classification. From the original set S, we define a set of partial trajectories S_P by sampling a subset of points from the original trajectories from 10% of the original length up to 100% in steps of 10. A partial trajectory is said to be misclassified if it is not assigned to the original cluster group based on the full trajectory set S. The classification is performed based on the input vectors consisting of

coefficients or PBF vectors. This is repeated for the SOM and *K*-Means defined set of codebook vectors. Assuming the size of the input vector to be *m*, the Euclidean distance is then calculated between the input vector B_i representing the partial trajectory and weight vector W_{ik} associated with *k*th output neuron as

$$D_k = \sqrt{\sum_{i=1}^{m} (B_i - W_{ik})^2} \qquad \forall k \tag{9}$$

The partial trajectory is then allocated to the class associated with the minimum value of D_k. If the class is different from the one to which the full trajectory was allocated then it is recorded as a misclassification. In the case of flow vector encoded trajectories, the point-based Euclidean distance measure is used. The percentage of misclassified trajectories taken over S_P is calculated for each method and for various sampling rates. The use of eq. (9) rather than *k*-NN classifier saves time as only the *m* weight vectors need be examined rather than the full test set. As evidenced from Fig. 7, the classifier derived from SOM in the coefficient subspace again outperforms *K*-Means. Furthermore, parameterized models prove more effective than point-based flow vectors in the trajectory prediction and classification task. These results give further impetus to the development of alternative dimensionality reduction techniques for learning and prediction of motion activity patterns.

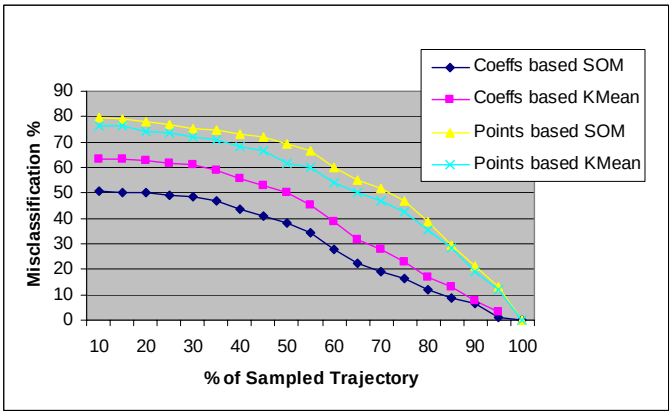
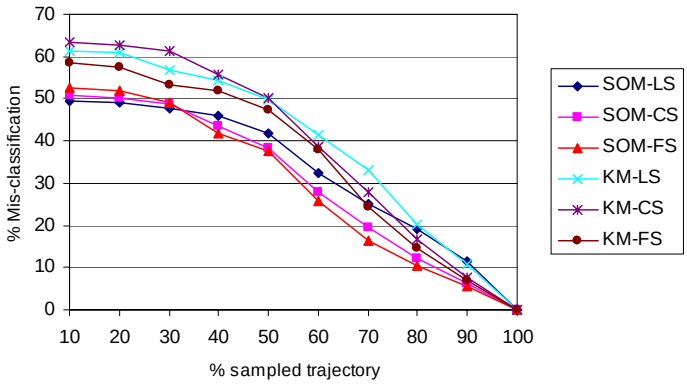


Fig. 7. Comparison of mean overall classification accuracy in trajectory prediction using 2 different clustering techniques (SOM and *K*-means) and 2 different trajectory encodings (coefficient subspace and discrete PBF vectors). #classes : #trajectories = 9 : 111.

Finally, we repeated the above experiment on trajectory prediction but compared the classification accuracy performance of the 3 approximations schemes for coefficient subspace learning using SOM and *K*-means. The use of SOM outperforms K-means in all cases of unsupervised learning and Fourier approximations records the highest classification accuracy over the partial sampling range 40-90%. The potential for predicting the correct spatiotemporal trajectory pattern from a partial trajectory has therefore been demonstrated.



6 Discussion and Conclusion

This paper presents a neural network learning algorithm for classifying spatiotemporal trajectories. Global features of motion trajectories are found to be represented well by polynomial and Fourier series approximations and this is apparent in the cluster visualizations. Using coefficients of basis functions as input feature vectors to a neural network learning algorithm offers an efficient alternative to the use of flow vectors for trajectory classification and prediction.

A current disadvantage is the representation of highly complex trajectories (e.g. hand trajectories in sign language) which are inherently unsuited to a global polynomial-based or Fourier representation. One possibility is to use a trajectory segmentation or multiscale approach and augment the feature vector with additional entries relating to object shape or colour. In future work we would like to extend this work to the autonomous detection of anomalous trajectories and prediction of unusual motion behaviour. A more comprehensive performance evaluation of other dimensionality reduction techniques in trajectory classification is also needed, e.g. ICA, HMMs and distance metric learning.

References

[1] S-F. Chang, W. Chen, H.J. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatiotemporal queries," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 602-615, Sept. 1998.

[2] S. Jeannin and A. Divakaran, "MPEG-7 visual motion descriptors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 720-724, June 2001.

[3] S. Dagtas, W. Ali-Khatib, A. Ghafor, and R.L. Kashyap, "Models for motion-based video indexing and retrieval," *IEEE Trans. Image Proc.*, vol. 9, no. 1, pp. 88-101, Jan 2000.

[4] Z. Aghbari, K. Kaneko, and A. Makinouchi, "Content-trajectory approach for searching video databases," *IEEE Trans. Multimedia*, vol. 5, no. 4, pp. 516-531, Dec. 2003.

[5] F. Bashir, A. Khokhar, and D. Schonfeld, "Segmented trajectory-based indexing and retrieval of video data," in *Proc. IEEE Int. Conf. Image Processing*, Spain, 2003, pp. 623-626.

[6] C-T. Hsu and S-J. Teng, "Motion trajectory based video indexing and retrieval," in *Proc. IEEE Int. Conf. Image Processing*, pt 1, 2002, pp. 605-608.

[7] F. Bashir, A. Khokhar, and D. Schonfeld, "A hybrid system for affine-invariant trajectory retrieval," in *Proc. MIR'04*, 2004, pp. 235-242.

[8] C. Shim and J. Chang, "Content-based retrieval using trajectories of moving objects in video databases," in *Proc. IEEE. 7th Int. Conf. Database Systems for Advanced Applications*, 2001, pp. 169-170.

[9] C. Shim and J. Chang, "Trajectory-based video retrieval for multimedia information systems," in *Proc. ADVIS*, LNCS 3261, 2004, pp. 372-382.

[10] Y. Jin and F. Mokhtarian, "Efficient video retrieval by motion trajectory", in *Proc. BMVC'04*, 2004.

[11] S. Khalid and A. Naftel, "Evaluation of matching metrics for trajectory-based indexing and retrieval of video clips," in *Proc.7th IEEE Workshop Applics. Comp. Vision (WACV/MOTION'05)*, Colorado, USA, Jan. 2005, 242-249.

[12] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585-601, 2003.

[13] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. Systems, Man & Cybernetic*, Part C, vol.34, no.3, pp. 334-352, August 2004.

[14] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," *Image Vis. Comput.*, vol. 14, no. 8, pp. 609-615, 1996.

[15] J. Owens and A. Hunter, "Application of the self-organising map to trajectory classification," in *Proc. IEEE Int. Workshop Visual Surveillance*, pp. 77-83, 2000.

[16] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank, "Traffic accident prediction using 3-D model-based vehicle tracking," *IEEE Trans. Vehicular Tech.*, vol. 53, no. 3, pp. 677-694, May 2004.

[17] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *Proc. IEEE CVPR*, June 2004.

[18] W. Hu, D. Xie, T. Tan, and S. Maybank, "Learning activity patterns using fuzzy self-organizing neural networks," *IEEE Trans. Systems, Man & Cybernetic*, Pt. B, vol. 34, no. 3, pp. 1618-1626, June 2004.

[19] R. Agrawal, C. Faloutsas, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms.*, 1993.

[20] K. Chan and A. Fu., "Efficient time series matching by wavelets," in *Proc. Int. Conf. Data Engineering*, Sydney, March 1999, pp. 126-133.
[21] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrota, "Locally adaptive dimensionality reduction for indexing large time series databases," in *Proc. ACM SIGMOD Conf.*, 2001, pp. 151-162.
[22] Y. Cui and R. Ng, "Indexing spatio-temporal trajectories with Chebyshev polynomials", in *Proc. ACM SIGMOD Conf.*, Paris, June 2004, pp. 599-610.
[23] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed.* Cambridge, England: Cambridge University Press, 1992.
[24] T. Kohonen, *Self-Organizing Maps*, 2nd ed. New York: Springer-Verlag, 1997, vol. 30.
[25] A.K Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1998.
[26] CAVIAR: [Online]. (2004, Jan. 10). [http://homepages.inf.ed.ac.uk/rbf/CAVIAR]