# MOTION CLUSTERING USING SPATIOTEMPORAL APPROXIMATIONS

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

## ABSTRACT

In this paper a new technique is proposed for the clustering and classification of spatio-temporal object trajectories extracted from video motion clips. The trajectories are represented as motion time series and modelled using Chebyshev polynomial approximations. Trajectory clustering is then performed to discover patterns of similar object motion. The coefficients of the basis functions are used as an input feature vector to a Self-Organising Map which can learn similarities between object trajectories in an unsupervised manner. It is shown that applying machine learning techniques in the Chebyshev parameter subspace leads to significant performance gains over previous approaches that encode trajectories as point-based flow (PBF) vectors. Experiments using the PETS'04 tracking dataset demonstrate the effectiveness of clustering in the parameter subspace and improvements in overall classification accuracy in comparison with PBF vector encoding. We also show how this technique can be further extended to the detection of anomalous motion paths. Applications to motion data mining and event detection in video surveillance systems are envisaged.

**Keywords**
motion data mining, trajectory clustering, classification, multimedia databases.

## 1. INTRODUCTION

Video surveillance and CCTV camera systems are now able to capture and store vast quantities of object motion data such as those of humans and vehicles However, techniques for reasoning about this data have so far failed to keep pace with developments in hardware. Content-based visual data management techniques are now urgently required for tasks such as motion data search and retrieval, discovery and grouping of similar motion patterns, detection of anomalous behaviour, motion understanding and prediction. In the next few years it is expected that intelligent surveillance systems incorporating such tasks will assume an increasingly important role in crime detection and prevention.

Much of the recent research focus has been on high-level object trajectory representation schemes that are able to produce compressed forms of motion data [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 a recent survey can be found in [12]. The literature on trajectory-based motion understanding and pattern discovery is less established 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 all been reported. Most of these techniques attempt to learn high-level motion behaviour patterns from sample trajectories using point-based flow vectors as input to the learning phase. In this paper, we show how to circumvent this requirement by proposing a trajectory classification approach based on a simple spatio-temporal representation scheme.

Related work within the data mining community on approximation schemes for modelling time series data is highly relevant to the parameterisation of motion trajectories. For example, spatio-temporal trajectories have been successfully modelled using discrete Fourier transforms (DFT) [19], wavelet transforms (DWT) [20], adaptive piecewise constant approximations (APCA) [21], and Chebyshev polynomials [22] although there are many others possibilities.

In this paper, we aim to apply time series modelling of spatio-temporal trajectories to the problem of trajectory classification and show how to learn motion patterns by projecting the high-dimensional trajectory data into a low-dimensional manifold represented by a suitably chosen parameter subspace. The vector of parameters is used as an input feature vector to a neural network learning algorithm - a SOM - which can learn similarities between object trajectories in an unsupervised manner. It is shown significant improvements in the accuracy of trajectory classification and recognition result whilst learning takes place in the parameter subspace rather than the original high-dimensional trajectory space. This is not suprising, e.g. many facial recognition systems have successfully exploited similar dimensionality reduction techniques.

The remainder of the paper is organized as follows. We review some relevant background material in section 2. In section 3, we present our trajectory representation approach. The algorithm for learning trajectories is then presented in section 4 within the framework of a SOM. In section 5, techniques for trajectory classification and prediction are discussed. The results of computer experiments with the PETS'04 pedestrian tracking datasets [26] are presented in section 6 in the context of motion trajectory clustering and classification. The paper concludes with a discussion of the advantages of our proposed technique over competing approaches and outlines prospects for further work.

## 2. REVIEW OF PREVIOUS WORK

Motion trajectory descriptors are known to be useful candidates for compressed representation of video object motion . Previous work has sought to represent moving object trajectories using simple piecewise linear or quadratic interpolation functions [1], [2], motion histograms [4] or discretised direction-based schemes [3], [8], [9]. Flexible spatio-temporal representations using piecewise polynomials were proposed by Hsu [6], although consistency in applying a trajectory-splitting scheme across query and searched trajectories has not been addressed. Affine and more general spatio-temporally invariant schemes have also been presented [5], [7], [10]. In addition to polynomial models, a wide variety of basis functions have been used to approximate more general spatio-temporal trajectories [19]-[22]. The importance of selecting the most appropriate trajectory model and similarity metric has received relatively scant attention in the literature [11]. Infact the choice of appropriate trajectory representation scheme can be highly domain dependent, e.g. hand trajectories arising from sign language are clearly more complex than vehicle motion trajectories in highway surveillance scenes.

It is surprising to find that many of these candidate spatiotemporal trajectory representation schemes have not yet been applied to the problem of motion data mining and trajectory classification. Recent work has either used probabilistic models such as HMMs [17] or point-based trajectory flow vectors [14], [16], [18] as a means of learning patterns of motion activity. Flow vectors consist of spatial coordinates augmented by instantaneous object velocities and possibly accelerations. These can be normalised to account for variation in trajectory lengths. An agglomerative clustering algorithm based on the Longest Common Subsequence (LCSS) approach is proposed in [13], [27] for grouping similar motion trajectories. The problem with PBF vector-encoded trajectory representation is the heavy computational burden making prospects for online learning of motion patterns remote.

The contribution of this paper is to show that by encoding trajectories as vectors of coefficients of approximating basis functions, these result in compact and stable input feature vectors to a learning algorithm which is used to classify motion patterns. Put another way, the technique implements a kind of manifold learning of trajectories in the parameter subspace.

## 3. TRAJECTORY REPRESENTATION

As it has been pointed out, there are many possibilities for a trajectory representation scheme. In view of the intended application, i.e. learning pedestrian motion patterns using video surveillance cameras, we compare several global modelling schemes - least squares polynomials, Chebyshev polynomial approximations and Fourier series. The latter two have been successfully used in time series data mining and least squares represents a simple straw-man approach. Complex spatio-temporal patterns can be modelled by considering the *x* and *y* projections as separate 1-D time series.

The output of an object tracking algorithm is normally a set of (noisy) 2-D 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 of the object's minimum bounding rectangle. The motion trajectory can be considered as comprising two independent 1-dimensional time series, $<t_i, x_i>$ and $<t_i, y_i>$, where $t_1 < \ldots < t_n$. The model for trajectory representation is based on least squares, Chebyshev polynomial approximations or Fourier series. High order orthogonal polynomials are appropriate for modelling complex spatiotemporal trajectories such as pedestrian tracking exhibiting stop-start and doubling-back motions.

### 3.1 Chebyshev Polynomials

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

$$[x \mid y] = f(t) \approx \sum_{k=0}^{m} b_k C_k \tag{1}$$

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

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

for $t \in [-1,1]$ and $i = 1,\ldots, m$. 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].

### 3.2 Least Squares and Fourier Series Approximations

Both least squares and Fourier series representations are well known approximation schemes and their details can be found elsewhere in standard numerical recipes texts [23]. In section 6, we compare the performance of the 3 methods in trajectory similarity retrieval under conditions of additive noise.

### 3.3 Similarity Search Metrics

The definition of similarity is central to the clustering and classification steps. Here a Euclidean distance function is used as the basis for comparing the similarity of motion trajectories. Each representation produces a vector of coefficients which can be used to index a 2-dimensional spatio-temporal trajectory in the parameter subspace. Given two trajectories *Q* and *S*, we can index these by a vector of 2(*m*+1) coefficients $Q = \left\{ \overrightarrow{q_0}, ...., \overrightarrow{q_m} \right\}$ and $S = \left\{ \overrightarrow{s_0}, ...., \overrightarrow{s_m} \right\}$, where $\overrightarrow{q_i}, \overrightarrow{s_i}$ are $\vec{q_i} = [q_{xi}, q_{yi}]^T$ and $\overrightarrow{s_i} = [s_{xi}, s_{yi}]^T$ ($i = 0,..., m$).

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

$$ED(Q,S) = \sqrt{\sum_{i=0}^{m} \left( \vec{q}_i - \vec{s}_i \right)^2} = \sqrt{\sum_{i=0}^{m} \left( q_{xi} - s_{xi} \right)^2 + \left( q_{yi} - s_{yi} \right)^2} \quad (3)$$

## 4. LEARNING TRAJECTORY PATTERNS USING SELF-ORGANIZING MAPS

Self-organised maps (SOMs) have been previously used for trajectory classification [15], [16] with trajectories encoded as point-based flow vectors. We propose to replace this step by parameter (coefficient) subspace encoding. A SOM discovers the underlying structure of motion trajectory data through unsupervised learning. Although it is usually described in the context of neural networks, it closely resembles a sophisticated clustering algorithm.

### 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-dimensional output layer. Each input neuron is connected to every output neuron with the connection represented by a weight vector. A similar architecture was used in [16] for learning vehicle trajectories as a means for accident prediction. 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 (or number of terms in the series) is determined empirically from the result of similarity retrieval experiments reported in section 6. In practice, there is little difference in performance once a minimum value of *m* is found. The number of output neurons represents the number of natural clusters *K* in the trajectory data and the value selected is the one generating the minimum Davies-Boulder (DB) cluster validity index [32].

### 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 in an agglomerative manner 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

$$\| B - W_c(t) \| \le \| B - W_k(t) \| \ \forall k \quad (4)$$

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

$$W_k(t+1) = W_k(t) + \alpha(t)\eta(k,c)(B - W_k(t)) \quad (5)$$

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

$$W_{ab} = \frac{mW_a + nW_b}{m+n} \quad (6)$$

## 5. TRAJECTORY CLASSIFICATION AND ANOMALY DETECTION

The SOM algorithm can be used to learn a set of motion patterns for the trajectory training dataset. The resulting labelled classes can then be used to classify unseen trajectories in the test set. We use a simple k-NN classifier with the optimum value of *k* chosen by means of leave-one-out crossvalidation. Classification results are presented in the following section using hand-labelled trajectories as ground truth.

Visualisation of the clusters in the parameter subspace shows that most classes can be approximated as Gaussian i.i.d. Anomalous trajectories can be detected through analysis of the covariance structure of a pattern at each output node. Those instance trajectories whose Mahalanobis distance from the cluster mean lies above a threshold are marked as anomalous..

During the SOM training phase, data is passed through the network and weights are adjusted. The training data is passed again through the network until each sample is allocated to a particular pattern (output node). The difference between the input vectors (representing a sample) and the weight vector of the pattern (to which it is allocated) is calculated for each sample. Having assumed that the class conditional probability density functions are normal, we can use the Mahalanobis distance (MD) as a plug-in classifier. If any sample vector exceeds a threshold on the MD to the closest pattern, the corresponding trajectory is considered abnormal. The Mahalanobis distance between the sample trajectory and codebook vector at each output node is then calculated. These values are then used to calculate the threshold for the difference of the instance with the closest pattern.

Assuming that sample $x$ belongs to cluster pattern $\Gamma_i$ , where $\#\{\Gamma_i\}$ denotes the number of vectors $x$ assigned to cluster $\Gamma_i$ and $i = 1,..K$. The cluster mean is denoted by $\mu_i$ and the covariance estimate is thus

$$\sum\nolimits_i = \sum\nolimits_{x\in\Gamma_i} (x-\mu_i)(x-\mu_i)^T \big/ \#\{\Gamma_i\} \tag{7}$$

where $\mu_i$ and $\Sigma_i$ are calculated during training. A trajectory is then considered abnormal if it satisfies the condition:

$$D_i = (x-\mu_i)^T(\Sigma_i)^{-1}(x-\mu_i) > c \tag{8}$$

where $D_i$ is the Mahalanobis distance of the input sample to the allocated pattern $i$ and $c$ is a tolerance threshold that determines the extent of deviation that is required to classify the behaviour as abnormal. Experiments that show the effect of varying $c$ on the detection of anomalous trajectories are shown in the following section.

## 6. EXPERIMENTS

We now present some results to indicate the effectiveness of the proposed trajectory clustering and classification technique. We evaluated the performance of the trajectory clustering algorithm using the CAVIAR tracking dataset used in PETS'04 [26]. The dataset consisted of hand annotated video sequences of moving and stationary people and is intended to provide a testbed for benchmarking vision understanding algorithms. Semantic descriptions of the target object behaviours and motion have been manually generated and stored in XML files. We have written a parser to extract the ground truth-labelled ($x$, $y$) coordinates of object trajectories. The dataset contains 222 trajectories as shown in Fig. 1.

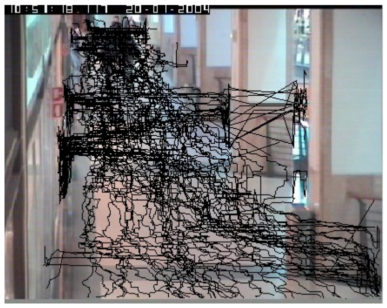

**Figure 1. Background scene containing database of ground truth labelled object trajectories.**

The performance of the three different trajectory representation schemes have first 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 size of the parameter set (i.e. vector of coefficients) used for representation and whether this has an effect on retrieval performance. From the original dataset $\Xi$, we produce a noise-corrupted dataset $\Xi_C$ by adding the term $\lambda * N(0,1) * rangeValues$ to each coordinate in the original set $\Xi$, where $0 \le \lambda \le 1$, $N$ is Gaussian noise of mean zero and variance 1, and *rangeValues* is the range on $X$ or $Y$ coordinates.

Each noisy trajectory in $\Xi_C$ is then selected as an example query $Q_C$ and we search for its closest match in the original dataset $\Xi$ by indexing the rank of the minimum distance $ED(Q_C,\Xi)$. A set of rankings $\forall\ Q_C \in \Xi_C$ is produced. In the absence of noise, the mean ranking should be 1 (i.e. each query should always return the distance to itself as zero and hence produce a ranking of 1). However, in the presence of noise this is not always the case, in particular when then are many similar trajectories to the query. 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 numbers of coefficients in LS, Chebyshev and Fourier Series approximations and for various values of $\lambda$. The results are summarised in Table 1.

The ED function in the parameter subspace is now used to perform the clustering stage. We have run the SOM algorithm using all the above approximation schemes and noticed that if the number of coefficients is too few, poor clustering results are obtained. This appears to rule out the use of LS representation. As a sanity check we used a standard $K$-means clustering algorithm and found similar results. It would appear that when using a small number of coefficients there is insuffcient discriminatory power in a very low dimensional subspace to achieve a meaningful clustering outcome.

| Approx. Scheme | $m$ | $\lambda = .1$ | $\lambda = .25$ | $\lambda = .75$ |
|---|---|---|---|---|
| Least Sq. | 1 | 97.3 | 93.7 | 76.1 |
| | 2 | 96.8 | 93.2 | 77.5 |
| | 3 | 97.3 | 92.8 | 75.2 |
| Chebyshev | 4 | 96.8 | 91.4 | 70.3 |
| | 8 | 97.3 | 92.3 | 71.6 |
| | 16 | 96.4 | 91.4 | 62.6 |
| | 32 | 96.8 | 88.7 | 55.0 |
| Fourier | 4 | 97.3 | 93.2 | 74.3 |
| | 8 | 96.8 | 91.4 | 66.2 |
| | 16 | 97.3 | 88.7 | 56.7 |

**Table 1. Percentage of noise corrupted trajectories correctly retrieved according to approximation scheme and noise level.**

In practice we have found that using Chebyshev approximations of degree 8 work well. Higher order approximations produce no discernible improvements to the clustering result. An attractive feature of orthogonal basis function approximations such as Chebyshev and Fourier series is that the choice of size of coefficient vector is not critical above a certain minimum ($m \approx 4$) provided that the dataset is not too noisy. In the presence of high noise contamination, a RANSAC approach is suggested.

We initially train a SOM network with 50 output neurons and then reduce these to 10 using the agglomerative clustering method described in section 4.2. The resulting trajectory groups are shown in Fig 2. Qualitatively similar motion trajectory patterns appear to have been grouped together quite successfully. For example, motions across the corridor from left-to-right and right-to-left are grouped into separate clusters as expected.

To investigate the effectiveness of clustering in the parameter subspace compared with with trajectories encoded as point-based flow vector, we performed some classification tests using proposed SOM technique and for comparison a standard *K*-means algorithm. The dataset $\Xi$ was randomly partitioned into training and test sets of equal sizes.

The trajectory patterns were learnt with both types of encoding (parameter and PBF vectors) using SOM and *K*-means methods on the training set. Using as ground truth class labels assigned by manual coding, we used a *k*-NN classifier (*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. The results summarised in Table 2 demonstrate the superiority of learning trajectory patterns in the parameter subspace. The classification accuracy using parameter subspace learning is higher than PBF vector encoding for both SOM and *K*-means algorithms.

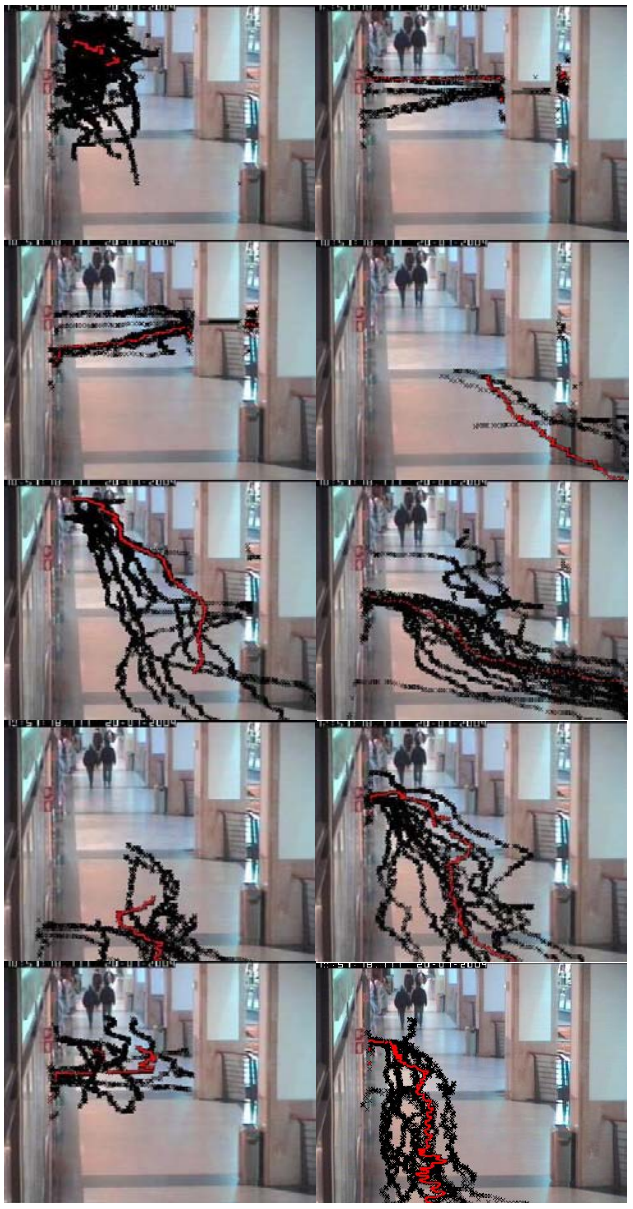
**Figure 2. Clustering spatiotemporal object trajectories using SOM. Trajectory closest to codebook vector is displayed in red.**

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

**Table 2. Comparison of mean overall classification accuracy for 4 different clustering techniques (#classes: # trajectories = 16 : 111).**

## 7. DISCUSSION AND CONCLUSION

This paper presents a neural network learning algorithm for classifying spatiotemporal trajectories. Global features of motion trajectories are represented well by polynomial approximations and this is apparent in the cluster visualizations. Using coefficients of basis functions as input vectors to a neural network learning algorithm offers a time-efficient alternative to the use of flow vectors for trajectory classification. Real-time learning of motion activity patterns for trajectory-based event detection and classification is thus made possible.

A current disadvantage is the handling of partial trajectory matching which is unsuited to a polynomial-based representation. One possibility is to parameterize the trajectory length and augment the feature vector with additional entries. In future work we would like to extend this approach to the autonomous detection of anomalous trajectories and prediction of unusual motion behaviour. A more comprehensive evaluation of other dimensionality reduction techniques for trajectory classification is also needed.

## REFERENCES

[1] Chang, S.F., Chen, W., Meng, H.J., Sundaram, H., and Zhong, D. A fully automated content-based video search engine supporting spatiotemporal queries. *IEEE Trans. Circuits Syst. Video Technol.*, 8, 5, 602-615, Sept. 1998.

[2] Jeannin, S. and Divakaran, A. MPEG-7 visual motion descriptors. *IEEE Trans. Circuits Syst. Video Technol.*, 11, 6, 720-724, June 2001.

[3] Dagtas, S., Ali-Khatib, W., Ghafor, A., and Kashyap, R.L. Models for motion-based video indexing and retrieval. *IEEE Trans. Image Proc.*, 9, 1, 88-101, Jan 2000.

[4] Aghbari, Z., Kaneko, K., and Makinouchi, A. Content-trajectory approach for searching video databases. *IEEE Trans. Multimedia*, 5, 4, 516-531, Dec. 2003.

[5] Bashir, F., Khokhar, A., and Schonfeld, D. Segmented trajectory-based indexing and retrieval of video data. In *Proc. IEEE Int. Conf. Image Processing (ICIP'03)*, Spain, 2003, 623-626.

[6] Hsu, C.-T. and Teng, S.-J. Motion trajectory based video indexing and retrieval. In *Proc. IEEE Int. Conf. Image Processing (ICIP'02)*, pt 1, 2002, 605-608.

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

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

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

[10] Jin, Y. and Mokhtarian, F. Efficient video retrieval by motion trajectory. In *Proc. British Machine Vision Conf. (BMVC'04)*, 2004, 667-676.

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

[12] Wang, L., Hu, W., and Tan, T. Recent developments in human motion analysis. *Pattern Recognition*, 36, 3, 585-601, 2003.

[13] Buzan D., Sclaroff S., Kollios G., "Extraction and Clustering of Motion Trajectories in Video," *InternationalConference on Pattern Recognition*, 2004.

[14] Johnson, N. and Hogg, D. Learning the distribution of object trajectories for event recognition. *Image Vis. Comput.*, 14, 8, 609-615, 1996.

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

[16] Hu, W., Xiao, X., Xie, D., Tan, T., and Maybank, S. Traffic accident prediction using 3-D model-based vehicle tracking. *IEEE Trans. Vehicular Tech.*, 53, 3, 677-694, May 2004.

[17] Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. Discovering clusters in motion time-series data. In *Proc. IEEE Conf. Comp. Vis. Patt. Recog. (CVPR'03)*, June 2003, 375-381.

[18] Hu, W., Xie, D., Tan, T., and Maybank, S. Learning activity patterns using fuzzy self-organizing neural networks. *IEEE Trans. Systems, Man & Cybernetic*-B, 34, 3, 1618-1626, June 2004.

[19] Faloutsas, C., Ranganathan, M., and Manolopoulos, Y. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Conf.*, 1994, 419-429.

[20] Chan, K. and Fu A. Efficient time series matching by wavelets. In *Proc. Int. Conf. Data Engineering*, Sydney, March 1999, 126-133.

[21] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrota, S. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD Conf.*, 2001, 151-162.

[22] Cui, Y. and Ng, R. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *Proc. ACM SIGMOD Conf.*, Paris, June 2004, 599-610.

[23] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed.. Cambridge University Press, Cambridge, England, 1992.

[24] Kohonen, T. *Self-Organizing Maps*, 2$^{nd}$ ed. Springer-Verlag, New York, 1997.

[25] CAVIAR: [Online]. (2004, Jan. 10). Available: [http://homepages.inf.ed.ac.uk/rbf/CAVIAR].

[26] Jain, A.K. and Dubes, R.C. *Algorithms for Clustering Data*, Prentice Hall, 1998.

[27] Vlachos M., Kollios G., Gunopulos D., "Discovering Similar Multidimensional Trajectories," *Proceedings of the International Conference on Data Engineering*, p. 673, 2002.