Motion Mining

Stan Sclaroff, George Kollios, Margrit Betke, and Romer Rosales

Boston University, Computer Science Dept., Boston MA 02215, USA
sclaroff@cs.bu.edu
http://www.cs.bu.edu/fac/sclaroff/

Abstract. A long-term research effort to support data mining applications for video databases of human motion is described. Due to the spatio-temporal nature of human motion data, novel methods for indexing and mining databases of time series data of human motion are required. Further, since data mining requires a significant sample size to accurately model patterns in the data, algorithms that automatically extract motion trajectories and time series data from video are required. A preliminary system for estimating human motion in video, as well as indexing and data mining of the resulting motion databases is described.

1 Introduction

In the last decade, there has been an explosive growth in the number of computer systems that gather data about human motion via video cameras, magnetic trackers, eye trackers, motion capture body suits and gloves, etc. These systems generate streams of 3D motion trajectories or other time series data about human motion that are used in computer human interfaces, computer animation and special effects, analysis of human biomechanics, and surveillance of human activity. Recently, new efforts have formed around the issue of creating archives of human motion data for use as "standard data sets" in the development of new algorithms in the computer science community, as well as for use in studies conducted by researchers from other disciplines (e.g., [22,33,35]).

As these datasets grow, there will be an opportunity to analyze this massive data archive to gain new insights that can be used to improve our understanding and models of human motion. Insights gained through *motion mining* could lead to improved methods for computer-assisted physical rehabilitation, occupational safety, and ergonomics, as well as improved methods for sports training, medicine, and diagnosis. Furthermore, motion mining could lead to improved computer vision and pattern recognition algorithms that are specially tuned to basic patterns or clusters found in human motion databases. It could also enable algorithms that automatically recognize anomalous motions because they are outliers when considered as part of the motion database.

Data mining has emerged as an important discipline in the database field during the last few years. Two reasons can be identified: (1) a huge amount of data is available today and (2) traditional approaches to analyze such data from statistics and machine learning are inadequate to cope with it. The goal of

M. Tucci (Ed.): MDIC 2001, LNCS 2184, pp. 16–30, 2001.

data mining is to efficiently find and describe structures and patterns in large datasets. These patterns are previously unknown and not stored explicitly in the database. Examples of data mining tasks include clustering, identifying patterns, and detecting outliers. Data mining of image databases or databases that store encodings of visual events has received only a small amount of attention [17,30].

Database and data mining methods can be used to discover patterns in databases of human motion data. Such data has a spatio-temporal aspect that must be dealt with, and therefore a major issue here is to develop methods for indexing and mining databases of motion trajectories and time series data. Another important problem is to automatically extract and analyze motion data given motion capture or video sequences. Another promising direction is to develop tracking/recognition algorithms that can learn from the clusters or patterns of motion found in data mining [40]. Through a tight coupling between computer vision and data mining modules, more reliable tracking and recognition algorithms can be achieved. In this paper, we describe a preliminary system for estimating human motion in video, trajectory-based encoding of human activity, as well as trajectory-based retrieval and data mining. This work represents the first step towards our long-term goal of an automatic system for mining databases of human motion data.

2 Related Work

In one approach to motion mining, we can assume that each object's motion is represented as a sequence of multidimensional points that we call a *trajectory*. For instance, the trajectory might consist of the position of the object centroid at each time step. The main reason for using this representation is its simplicity and generality: every motion pattern can be represented as a time series of points moving in a low-dimensional space. Another reason is that simple representations will allow the design of more efficient and robust algorithms. This is very important when working with large datasets. Given trajectories represented in this way, a method for measuring similarity between trajectories is needed.

Perhaps the simplest approach to define the similarity between two sequences is to map each sequences into a vector and then use a p-norm distance to define the similarity measure. The p-norm distance between two n-dimensional vectors $\bar{x}$ and $\bar{y}$ is defined as $L_p(\bar{x}, \bar{y}) = (\sum_{i=1}^{n}(x_i - y_i)^p)^{\frac{1}{p}}$. For $p = 2$ is the well know Euclidean distance and for $p = 1$ the Manhattan distance. The advantage of this simple model is that it allows efficient indexing by a dimensionality reduction technique [1,48,19,15]. On the other hand the model cannot deal well with outliers and is very sensitive to small distortions in the time axis. There are a number of interesting extensions to the above model to support various transformations such as scaling [12,37], shifting [12,21], normalization [21] and moving average [37]. Other recent works on indexing time series data for similarity queries assuming the Euclidean model include [27,26].

Another approach is based on the time warping technique that first has been used to match signals in speech recognition [41]. Berndt and Clifford [3] proposed

to use this technique to measure the similarity of time-series data in data mining. The idea is to allow stretching in time in order to get a better distance. Recently, indexing techniques for this similarity measure have been proposed [28,34].

Other techniques to define time series similarity extract certain features (Landmarks [36] or signatures [14]) from each time-series and then use these features to define the similarity. An interesting approach to represent a time series using the direction of the sequence at regular time intervals is presented in [46]. Ge and Smyth [18] present an interesting alternative approach for sequence similarity that is based on probabilistic matching. A domain independent framework for defining queries in terms of similarity of objects is presented in [25].

Note that all the above work deals mainly with one dimensional time-series. An approach to indexing two-dimensional moving object trajectories in video databases was proposed in [11]. The system also provides a sketched-based user interface for formulating trajectory-based queries, but provides no data mining component. Moving blobs are automatically segmented, given a motion-stabilized video sequence as input. Moving blobs are indexed using color and/or texture features. In addition, the series of positions of a blob's centroid in each video frame is stored as a motion trail. This system employs variants of the Euclidean distance metric to enable time-normalized retrieval of similar trajectories. As mentioned earlier, the Euclidean distance measure is sensitive to outliers, and nonlinear distortions of the time axis.

The most related paper to our work is the Bozkaya, et al. [7]. They discuss how to define similarity measures for sequences of multidimensional points using a restricted version of the edit distance. Also, they present two efficient methods to index the sequences for similarity retrieval. However, they focus on sequences of feature vectors extracted from images and not trajectories and they do not discuss transformations or approximate methods to compute the similarity. In another recent work, Lee et al. [31] propose methods to index sequences of multidimensional points. They extend the ideas presented by Faloutsos et al. in [16] and the similarity model is based on the Euclidean distance.

A recent work that proposes a method to cluster trajectory data is due to Gaffney and Smyth [17]. They use a variation of the EM (expectation maximization) algorithm to cluster small sets of trajectories. However, their method is a model based approach that usually has scalability problems. Also, it implicitly assumes that the data (trajectories) follow some basic models which are not easy to find and describe in real datasets.

3 Estimating Human Motion Trajectories

Spatio-temporal indexing of trajectories relies on algorithms that can detect, estimate, and encode relevant information about human motion in image sequences. Since data mining requires a significant sample size to accurately model patterns in the data, algorithms that automatically extract motion trajectories and time series data from video are required. Therefore, a first concern in building our system will be detecting and segmenting changing or moving blobs in video.

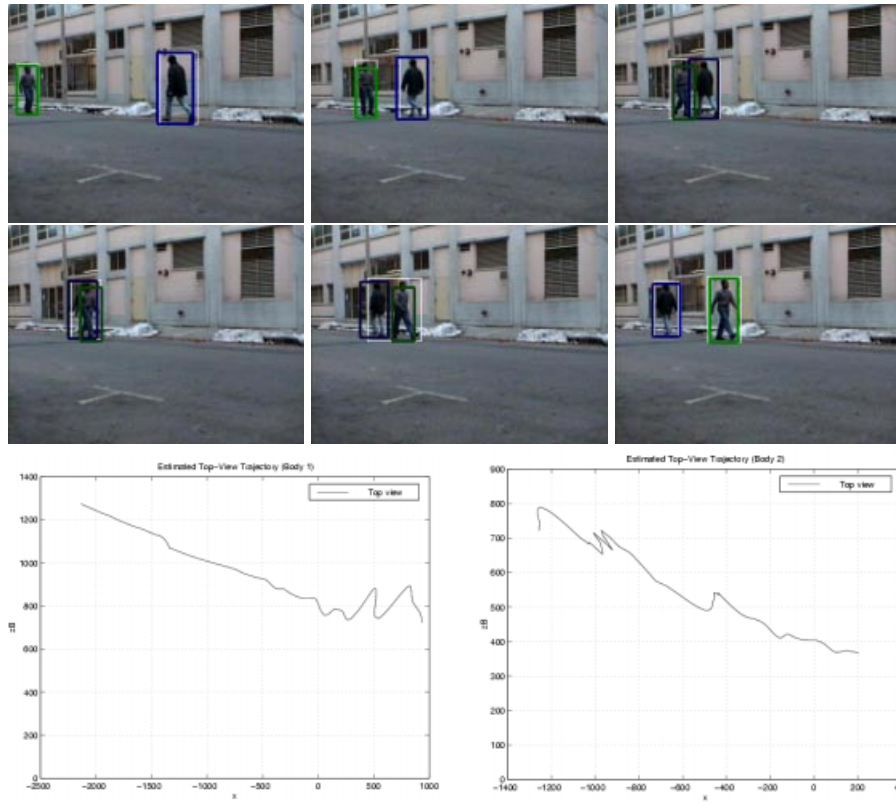

Fig. 1. Trajectory estimation example: Two bodies walking along different trajectories, occluding each other. The estimated minimum bounding boxes (MBRs) for each moving object are shown overlaid on the input video images. The graphs show the recovered trajectories (top view) for the two moving bodies. Note that motion of Body 1 is from right to left. Body 2 goes left to right. Trajectory estimates improve over time, as the extended Kalman Filter converges given more video frames.

We will assume that the time-varying image sequences have been registered and rectified to correct for motion of the camera, as well as normalized for differences in imaging conditions. Given a set of registered images, we can make use of change detection and moving blob segmentation methods that rely on first and second order statistics [39,47] or adaptive mixture models [43]. A connected components analysis is then applied to the resulting image. Initial segmentation is usually noisy, so morphological operations and size filters are applied.

To estimate the motion trajectory for each blob, we can use a predictive tracker [39], which is based on a first order Extended Kalman Filter (EKF) [42]. The EKF has proven to be very useful in recovery of rigid motion and structure from image sequences [9,2,8,38]. Most of these approaches assume rigid motion. One of the first important results on recursive structure and motion

estimation was the work of [9]. The formulation of [2] yields improved stability and accuracy of the estimates. In both methods, image feature tracking and correspondence are assumed. In this paper, we present a method that automatically tracks multiple moving objects, and use this information to estimate 3D translational trajectories (up to a scale factor).

To model trajectories, [8] assumed that the surface on which the motions occur was known, and also that this surface was a plane. Each object was represented as a point moving in the plane, partially avoiding problems related to changes in shape. It is also possible to reducing tracking to a plane, if the projective effect is avoided through the use of a top, distant view [24]. It is also possible to use some heuristics about body part relations and motion on image plane like [23]. In our work, we do not assume planar motion or detailed knowledge about the object and our formulation can handle some changes in shape.

In our approach, each blob's tracker T_i contains information about object location, a binary support map, blob characteristics, minimum bounding rectangle (MBR), etc. For this application, we can choose an EKF state $\mathbf{x}$ that models the blob's MBR moving along a piece-wise linear trajectory:

$$\mathbf{x} = (x_0, y_0, x_1, y_1, z\beta, \dot{x}_0, \dot{y}_0, \dot{x}_1, \dot{y}_1, \dot{z\beta}). \tag{1}$$

In the state vector (x_0, y_0) and (x_1, y_1) are the corners of the MBR, z is the relative distance from the camera, and $\beta = \frac{1}{f}$ is the inverse camera focal length. Note that if the focal length is unknown, this formulation does not provide a unique solution in 3D space. However, the family of allowable solutions all project to a unique solution on the image plane. We can therefore estimate objects' future positions on the image plane and their image trajectories given their motion in $(x, y, z\beta)$ space. It is therefore assumed that although the object to be tracked is highly non-rigid, the 3D size of the object's bounding box will remain approximately the same, or at least vary smoothly. This assumption might be too strong in some cases; *e.g.*, if the internal motion of the object's parts cannot be roughly self contained in a bounding box. However, when analyzing basic human locomotion, we believe that these assumptions are a fair approximation.

For our representation a 3D central projection model similar to [44,2] is used:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \frac{1}{1 + z\beta}, \tag{2}$$

where (x, y, z) is the real 3D feature location in the camera reference frame, (u, v) is the projection of it to the camera plane, and $\beta = \frac{1}{f}$ is the inverse focal length. The origin of the coordinate system is fixed at the image plane. This model has proven to be useful when estimating focal length and structure in the structure from motion problem [2]. One important property of this model is that it is numerically well defined even in the case of orthographic projection.

Our EKF process is guided by the following linear difference equation:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k, \tag{3}$$

where $\mathbf{x}_k$ is our state at time k, $\mathbf{w}_k$ is the process noise and $\mathbf{A}_k$, the system evolution matrix, is based on first order Newtonian dynamics and assumed time invariant ($\mathbf{A}_k = \mathbf{A}$). If additional prior information on dynamics is available, then $\mathbf{A}$ can be changed to better describe the system evolution [38]. In our case, we use the assumption that trajectories are locally linear in 3D.

Our measurement vector is $\mathbf{z}_k = (u_{0k}, v_{0k}, u_{1k}, v_{1k})$, where u_{ik}, v_{ik} are the image plane coordinates for the observed feature i at time k. The measurement vector is related to the state vector via the measurement equation: $\mathbf{z}_k = h(x_k + v_k)$. Measurement noise is assumed to be additive in our model. The EKF time update equation becomes:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}_k \hat{\mathbf{x}}_k \tag{4}$$

$$\mathbf{P}^-_{k+1} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{W}\mathbf{Q}_k \mathbf{W}^T \tag{5}$$

where $\mathbf{A}$ represent the system evolution transformation, $\mathbf{Q}_k$ is the process noise covariance. The matrix $\mathbf{W}$ is the Jacobian of the transformation $\mathbf{A}$ with respect to $\mathbf{w}$. Finally, the measurement update equations become:

$$\mathbf{K}_k = \mathbf{P}^-_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}^-_k \mathbf{H}_k^T + \mathbf{V}\mathbf{R}_k \mathbf{V}^T)^{-1} \tag{6}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_\mathbf{k}{}^-, 0)) \tag{7}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}^-_k, \tag{8}$$

where $\mathbf{H}_k$ is the Jacobian of $h(\bullet)$ with respect to the estimate of $\mathbf{x}$ at time k:

$$\mathbf{H}_k = \begin{bmatrix} \frac{1}{\lambda} & 0 & 0 & 0 & -\frac{x_0}{\lambda^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 & 0 & -\frac{y_0}{\lambda^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 0 & -\frac{x_1}{\lambda^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\lambda} & -\frac{y_1}{\lambda^2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{9}$$

where $\lambda = 1 + z\beta$. The matrix $\mathbf{V}$ is the Jacobian of $h(\bullet)$ with respect to $\mathbf{v}$, and $\mathbf{R}_k$ is our measurement noise covariance at time k. In this formulation, the general assumptions are: $\mathbf{w}$ is a Gaussian random vector with $p(\mathbf{w}_k) \sim N(0, \mathbf{W}\mathbf{Q}_k\mathbf{W}^T)$, and $\mathbf{v}$ is also Gaussian $p(\mathbf{v}_k) \sim N(0, \mathbf{V}\mathbf{R}_k\mathbf{V}^T)$. For more detail, see [39,42].

Another problem is occlusion, a problem that cannot be ignored in the segmentation of multiple moving objects. Occlusion can occur when another object partially (or completely) obstructs a camera's view of the human; e.g., two people's paths cross in the image yielding a temporary occlusion of one person from that camera's viewpoint. Therefore, any general system must include algorithms that can reliably detect and maintain the tracking of moving objects before, during, and after an occlusion. Fortunately, given our EKF-based approach, occlusion time can be estimated using the EKF predictions and estimates of MBR's velocity and position.

3.1 Estimating Trajectories of Human Body Components

Estimation of trajectories of locomotion is one possible source of human motion databases. Trajectories of different components of the human body are also

important in applications ranging from computer human interfaces to motion studies. Methods for automatic and robust detection, tracking, and interpretation of human body components and their motion in video under normal lighting conditions have been developed [4,5,20]. The goal is to reliably estimate the motion trajectories of, for example, a finger, foot, or facial feature in real-time and interpret them as a communication of the computer user.

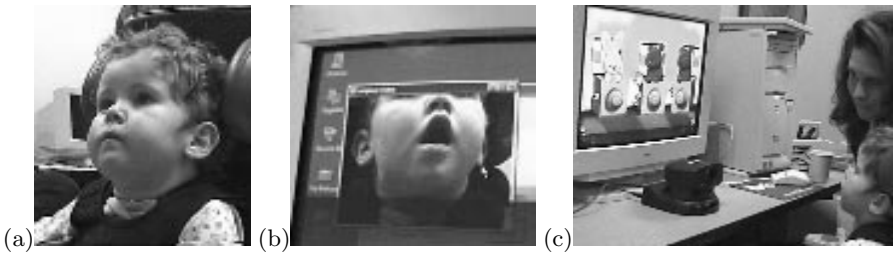

Fig. 2. (a) Thirty-month old *Camera-Mouse* user with cerebral palsy. (b) The vision system's view of the user. (c) The user playing with educational software (video camera is below the user computer's monitor).

One application of this approach is found in the "Camera Mouse" system [20], which has been developed to provide computer access for people with severe disabilities. The system tracks the computer user's movements with a video camera and translates them into the movements of the mouse pointer on the screen. Fig. 2 shows a thirty-month old user of the Camera Mouse system and her tracked face on the monitor of the vision computer. Here the vision algorithm is tracking her lower lip.

In Fig. 3 we show an example of three trajectories of mouse pointer movements that correspond to human facial movements tracked with Camera Mouse system. These trajectories were obtained from a non-handicapped user of the system. The Camera Mouse was used with a spelling program. The three trajectories were created when the computer user spelt out some words by moving the mouse pointer to an area on the screen that corresponds to a particular letter and selecting the letter ("clicking" the mouse) by lingering over the area for about a second. The figure represents the output of vision algorithms (data sets) used in evaluating preliminary versions of a data mining system.

4 Motion Mining

Given various types of human motion trajectories, our attention now returns to the issue of data mining. Many data mining tasks require a similarity model (or a distance function) for the objects stored in the database. The problem is non-trivial. Most of the current methods in the data mining community are based on mapping the time-series with n elements to a vector in an n-dimensional space.

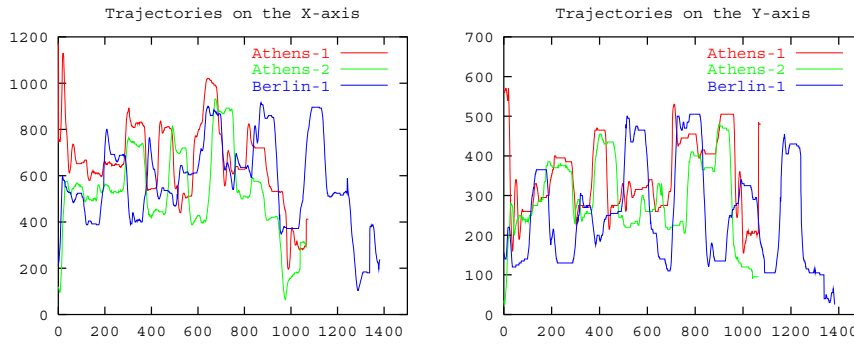

Fig. 3. Trajectories of mouse pointer movements that correspond to human facial movements tracked with Camera Mouse system. The motion on X and Y axis during the spelling of two words – Athens (twice) and Berlin (once) are shown.

They then use a p-norm distance function to define the similarity measure. As explained in Sec. 2, the p-norm is inadequate to deal with trajectories.

A general way to address these issues is to use the Longest Common Sub-Sequence ($LCSS$) model [6]. The basic idea is to try to match two sequences by allowing them to stretch without rearranging the sequence of values. The $LCSS$ is a variation of the Time Warping model [3,41], which has been proved very effective in other domains, for example in speech and gesture recognition, robotics, medicine, etc. In addition, the $LCSS$ model is more robust than time warping with respect to outliers. Next we present some simple similarity models and then we proceed with a discussion on how to use these model to index and cluster sets of trajectories. For this discussion, we assume that objects are points that move on the (x, y)-plane and time is discrete. However, our techniques are general and can be applied to objects moving in an n-dimensional space.

Let $A = ((a_{x,1}, a_{y,1}), \ldots, (a_{x,n}, a_{y,n}))$ and $B = ((b_{x,1}, b_{y,1}), \ldots, (b_{x,m}, b_{y,m}))$ be two trajectories of moving objects with size n and m respectively. For a trajectory A, let $Head(A) = ((a_{x,1}, a_{y,1}), \ldots, (a_{x,n-1}, a_{y,n-1}))$.

Definition 1. *Given an integer δ and a real number $0 < \epsilon < 1$, we define the $LCSS_{\delta,\epsilon}(A, B)$ as follows:*

$$LCSS_{\delta,\epsilon}(A,B) = \begin{cases} 1 + LCSS_{\delta,\epsilon}(Head(A), Head(B)) & \text{if } |a_{x,n} - b_{x,m}| < \epsilon \text{ and} \\ & \quad |a_{y,n} - b_{y,m}| < \epsilon \text{ and} \\ & \quad |n - m| \leq \delta \\ \max(LCSS_{\delta,\epsilon}(Head(A), B), & \\ \quad LCSS_{\delta,\epsilon}(A, Head(B))) & \text{otherwise} \end{cases}$$

The constant δ controls how far in time we can go in order to match a given point from one trajectory to a point in the another trajectory. The constant ϵ is the match threshold.

Definition 2. *We define the similarity function* $S1$ *between two trajectories* A *and* B*, given* δ *and* ϵ*, as follows:*

$$S1(\delta, \epsilon, A, B) = \frac{LCSS_{\delta,\epsilon}(A, B)}{\min(n, m)}$$

This first similarity function is based on the $LCSS$ and the idea is to allow time stretching. Then, objects that are close in space at different time instants can be matched if the time instants are also close.

We use this function to define two new similarity measures that are more suitable for trajectories. The first one is based on approximating the trajectory with a signature, which is similar to the Landmarks model [36]. Given a time period τ we compute the location of the object every τ time instants (0, τ, 2τ, etc.) Then, we approximate the trajectory with a piecewise-linear curve by connecting the consecutive points with a line segment. Next, we compute the direction of each line segment by projecting the segment on to the (x, y)-plane and find the angle of the segment with respect to the x-axis. If ϕ is the angle of a segment s and $i * \frac{2\pi}{k} \leq \phi < (i+1) * \frac{2\pi}{k}$, we replace the segment s with the value i. Doing the same for every segment, we get a sequence of symbols, one for each segment. Thus, we represent the trajectory of a moving object as a sequence of symbols. We only need $k+1$ symbols, one for each angle range and one more to represent static object. We call the above sequence the *motion signature* of the trajectory. An example of a motion signature for a trajectory is shown in Fig. 4. In the example k is equal to 4. Using the motion signatures we now define the similarity of two trajectories by computing the $LCSS$ of the signatures.

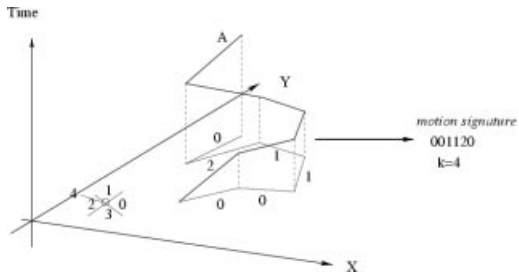

Fig. 4. Motion Signature of A.

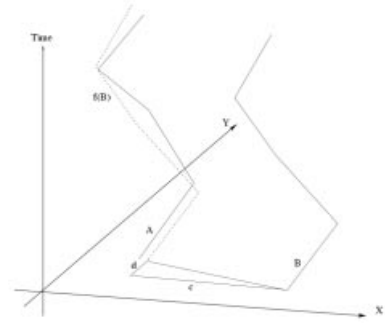

Fig. 5. Translation of Trajectory B.

Definition 3. *Let* A *and* B *be two trajectories with motion signatures* $A' = (a'_1, \ldots, a'_{n'})$ *and* $B' = (b'_1, \ldots, b'_{m'})$*, where* $a'_i, b'_j \in (0, \ldots, k)$*. Then, given an*

integer δ we define the similarity function $S2$ as follows:

$$S2(\delta, A, B) = \frac{LCSS_\delta(A', B')}{\min(|A'|, |B'|)}$$

where,

$$LCSS_\delta(A', B') = \begin{cases} 1 + LCSS_\delta(Head(A'), Head(B')) & \text{if} a'_{n'} = b'_{m'} \text{ and} \\ & |n' - m'| \leq \delta \\ \max(LCSS_\delta(Head(A'), B'), & \\ \quad LCSS_\delta(A', Head(B'))) & \text{otherwise} \end{cases}$$

Using the above method we can detect objects with similar movements even if these objects move in different locations and their movement is not synchronized. Another interesting point is that we can use only $log_2 k$ bits to represent each symbol and the signature is usually much smaller in size than the original trajectory. Therefore, the computation of the similarity is very efficient.

To create a more accurate measure, we define similarity based on exact trajectories. First, we consider the set of translations. A translation simply shifts a trajectory in space by a different constant in each dimension. Let $\mathcal{F}$ be the family of translations. Then a function $f_{c,d}$ belongs to $\mathcal{F}$ if $f_{c,d}(A) = ((a_{x,1} + c, a_{y,1} + d), \ldots, (a_{x,n} + c, a_{y,n} + d))$. Next, we define a second notion of the similarity based on the above family of functions.

Definition 4. *Given δ, ϵ and the family $\mathcal{F}$ of translations, we define the similarity function $S3$ between two trajectories A and B, as follows:*

$$S3(\delta, \epsilon, A, B) = \max_{f_{c,d} \in \mathcal{F}} S1(\delta, \epsilon, A, f_{c,d}(B))$$

By allowing translations, we can detect similarities between movements that are parallel in space, but not identical. In addition, the $LCSS$ model allows stretching and displacement in time, so we can detect similarities in movements that happen with different speeds, or at different times. In Fig. 5 we show an example where a trajectory B matches another trajectory A after a translation is applied. Note that the value of parameters c and d are also important since they give the distance of the trajectories in space. That can be a useful information when we analyze trajectory data.

To compute the similarity functions $S1$ and $S2$ we have to run a $LCSS$ computation to the two sequences. The $LCSS$ can be computed by a dynamic programming algorithm in $O(n^2)$ time. However, we only allow matchings when the difference in the indices is at most δ, and this allows the use of a faster algorithm. We can show that given two trajectories A and B, with $|A| = n$ and $|B| = m$, we can compute the $S1$ and $S2$ distances in $O(\delta(n + m))$ time. For the $S3$ distance we use an approximation algorithm that can find the distance between two trajectories A and B with error smaller than β in $O((m+n)\delta^3/\beta^2)$ time. Given trajectories A, B with lengths n, m respectively, and constants δ, β, ϵ, the approximation algorithm is as follows:

1. *Using the projections of A, B on the two axes, find the sets of all different translations on the x and y axis.*
2. *Find the $i\frac{\beta(n+m)}{2}$-th quantiles for each set, $1 \leq i \leq \frac{4\delta}{\beta}$.*
3. *Run the $LCSS_{\delta,\epsilon}$ algorithm on A and B, for each of the $(\frac{4\delta}{\beta})^2$ pairs of translations.*
4. *Return the highest result.*

4.1 Indexing Trajectories of Moving Objects

Given some distance measure, we need to design new index methods to store and retrieve trajectories of moving objects using this method. Indexing methods play a very important role in exploratory data mining, where the analyst makes hypotheses about the data and asks queries for validation. Considering the size of the datasets and the on-line nature of the analysis, a system with no indexing capabilities will be of limited use.

An important observation for the $S2$ distance measure is that for trajectories of equal size it is actually a metric (that is, the triangle inequality holds). A metric distance function is important for indexing, because we can prune a large part of the dataset during the query phase based on the triangle inequality. Therefore, our method is to divide first the different trajectories into groups of equal (or almost equal) length and then index each group separately. One approach is to use indexing methods for general metric spaces, e.g., the M-tree [13] or the vp-tree [45].

Another approach to index a set of trajectories is to embed them into a normed space $\mathcal{D}$ and try to keep the pairwise distances as close as possible to the original ones. Ideally, $\mathcal{D}$ will be a low dimensional Euclidean space $\Re^d$, where d is small. An interesting embedding method is presented in [32]. The basic idea is to select a set of subsets of S. Let X be a subset of S. Then we find the minimum distance of a given trajectory t to X, $D(t, X) = min_{x \in X} d(x, t)$. This number defines the coordinate of t for the dimension that corresponds to X. Using d number of subsets, $X_1, X_2, \ldots, X_d$, we map each trajectory $t \in S$ to a vector $[D(t, X_1), D(t, X_2), \ldots, D(t, X_d)]$. The distance between two vectors is defined using the l_1 or l_2 norm. However, the problem with this approach is that the distortion in the pairwise distances can be large and we may have many false negatives.

For the $S3$ distance, indexing is more challenging since this measure is not a metric. However, we can still use the embedding approach and get approximate answers. Another alternative is to cluster the set of trajectories and then use the clusters to answer nearest neighbor queries. That is, given the clusters and a query trajectory, find the clusters that have a representative point closer to the query trajectory and then report the trajectories in these clusters, possibly sorted by the distance to the query.

Another type of useful query is the *subsequence match query*, where we specify a part of a trajectory S_p and we ask the system to find the trajectories that have a similar subsequence. To solve this problem we can partition the original trajectories into smaller ones and we can index these subsequences.

4.2 Motion Clustering and Outlier Detection

A very important data mining task is to cluster set of objects in a large dataset. Therefore, it is important to design clustering methods for large sets of trajectories using various distance functions. The results of a clustering task can be used directly to characterize different groups of objects and summarize their main characteristics. The clusters will be used for re-training the classification and prediction algorithms in the Computer Vision sub-system. Also, hierarchical clustering algorithms can be used for indexing large datasets for similarity queries as we mentioned above.

One of the few methods to cluster trajectory data taking into account the special properties of trajectories has been proposed in [17]. They use a variation of the EM (expectation maximization) algorithm to cluster small sets of trajectories. A problem with this approach is its scalability. Also, the distance measure is based on the probabilistic model which may not be the most appropriate for some specific applications.

Another important task in a data mining system is to identify outliers. An outlier is an object that behaves in an unusual and unpredictable manner. Outlier mining has been used in fraud detection, by detecting unusual usage of credit cards or telecommunication services [10]. In our type of applications we are interested to find unusual or strange motion patterns. Actually, these patterns are sometimes more interesting for further analysis.

The first issue in outlier detection is to define what data is considered as an outlier for a given dataset. The statistical approach to define outliers is to assume that the distribution of the objects in the dataset follows a specific model and then try to identify objects that deviate from this model. Unfortunately, finding an appropriate model for datasets of trajectories is very difficult and usually real datasets do not follow general statistical models. Another definition of outliers uses a different approach that extends the distance based outlier definition by [29]. In particular, given a function that describes a distance between any two objects in the database, we say that an object O is a $DT(k, \xi)$ outlier, if there are at most k objects in the database the have a distance to O smaller than ξ. The challenge then is to find efficiently all the outliers, given some values for k and ξ. An alternative approach is to find the distance of each object to its k-th closest object and report a list of the objects ordered by this distance. Clearly, objects that are "far" from the others will appear first in the list.

5 Conclusion

We have described preliminary work towards a *motion mining* system for content-based retrieval and pattern discovery in video databases of human motion. The proposed methods are tailored to capture the spatio-temporal nature of human motion data. Two approaches to automatic estimation of human motion trajectories were described and tested. Since data mining requires a significant sample size to accurately model patterns in the data, algorithms that automatically

extract motion trajectories and time series data from video are required. We defined trajectory representations that can be used in computing similarity between motion trajectories based on the LCSS method. This preliminary system enables ongoing and future work in the areas of spatio-temporal indexing, clustering, and outlier detection for large databases of human motion. A key issue in the immediate future will be evaluating the retrieval and mining methods on large video databases.

In future work, we plan to investigate alternative representations for trajectories. In particular, the use of a probabilistic model, e.g., Hidden Markov Models (HMMs) or Semi-Markov Models [18]), seems like a promising direction to pursue. However, in this case distance measures needed for indexing, mining, and retrieval are more difficult to compute in an efficient manner given current techniques. The embedding approaches described in Sec. 4.1 should prove quite useful in this regard.

Acknowledgments

This work was supported in part through US ONR Young Investigator Award N00014-96-1-0661, and NSF grants IIS-9624168 and EIA-9623865.

References

1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. *In Proc. of the 4th FODO*, pages 69–84, October 1993.
2. A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 17(6), 1995.
3. D. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. *In Proceedings of KDD Workshop*, 1994.
4. M. Betke and J. Kawai. Gaze detection via self-organizing gray-scale units. In *Proc. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 1999.
5. M. Betke, B. Mullally, and J. Magee. Active detection of eye scleras in real time. In *Proc. IEEE CVPR Workshop on Human Modeling, Analysis and Synthesis*, 2000.
6. B. Bollobas, G. Das, D. Gunopulos, and H. Mannila. Time-Series Similarity Problems and Well-Separated Geometric Sets. *In Proc of the 13th SCG, Nice, France*, 1997.
7. T. Bozkaya, N. Yazdani, and M. Ozsoyoglu. Matching and Indexing Sequences of Different Lengths. *In Proc.of the Intern. Conf. on Information and Knowledge Management, Las Vegas*, 1997.
8. K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3D motion trajectories and their use in prediction. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 19(3), 1997.
9. T. Broida and R. Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 13(6):497-513, 1991.
10. M. Cahill, F. Chen, D. Lambert, J. Pinheiro, and Don Sun. *Detecting Fraud in the Real World.* Klewer, 2000.

11. S.F. Chang, W. Chen, J. Meng, H. Sundaram, and D. Zhong. A fully automated content based video search engine supporting spatio-temporal queries. *IEEE Trans. on Circuits and Systems for Video Technology*, 8(5), 1998.
12. K. Chu and M. Wong. Fast Time-Series Searching with Scaling and Shifting. *ACM Principles of Database Systems*, pages 237–248, June 1999.
13. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *In Proc. of the 23rd Conference on Very Large Data Bases, Athens, Greece*, pages 426–435, August 1997.
14. C. Faloutsos, H.V. Jagadish, A. Mendelzon, and T. Milo. Signature technique for similarity-based queries. *In SEQUENCES 97*, 1997.
15. C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *In Proc. ACM SIGMOD*, pages 163–174, May 1995.
16. C. Faloutsos, M. Ranganathan, and I. Manolopoulos. Fast Subsequence Matching in Time Series Databases. *In Proceedings of ACM SIGMOD*, pages 419–429, May 1994.
17. S. Gaffney and P. Smyth. Trajectory Clustering with Mixtures of Regression Models. *In Proc. of the 5th ACM SIGKDD, San Diego, CA*, pages 63–72, August 1999.
18. X. Ge and P. Smyth. Deformable Markov model templates for time-series pattern matching. *In Proc ACM SIGKDD*, 2000.
19. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. *In Proc. of 25th VLDB*, pages 518–529, 1999.
20. J. Gips, M. Betke, and P. Fleming. The camera mouse: Preliminary investigation of automated visual tracking for computer access. In *Annual Conf. of the Rehabilitation Eng. and Assistive Tech. Soc. of North America (RESNA)*, July 2000.
21. D. Goldin and P. Kanellakis. On Similarity Queries for Time-Series Data. *In Proceedings of CP'95, Cassis, France*, September 1995.
22. J. Hodgins. Digital Muybridge: A Repository for Human Motion Data. `http://www.interact.nsf.gov/cise/abst.nsf/awards/0079060`, May 2000.
23. L. Davis I. Haritaouglu, D. Harwood. W4s: A realtime system for detecting and tracking people in 2.5d. In *Proceedings of Third European Conference on Computer Vision*, 1998.
24. S. Intille and A. F. Bobick. Real time close world tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1997.
25. H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. *In Proc. of the 14th ACM PODS*, pages 36–45, May 1995.
26. T. Kahveci and A. K. Singh. Variable length queries for time series data. In *Proc. of IEEE ICDE*, pages 273–282, 2001.
27. E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. of ACM SIGMOD*, pages 151–162, 2001.
28. E. Keogh and M. Pazzani. Scaling up Dynamic Time Warping for Datamining Applications. *In Proc. 6th Int. Conf. on Knowledge Discovery and Data Mining, Boston, MA*, 2000.
29. E. Knorr and R. Ng. Algorithms for Mining Distance Based Outliers in Large Databases. *In Proceedings of VLDB, New York*, pages 392–403, August 1998.
30. G. Kollios, S. Sclaroff, and M. Betke. Motion mining: Discovering spatio-temporal patterns in databases of human motion. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, May 2001.

31. S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity Search for Multidimensional Data Sequences. *In Proceedings of ICDE*, pages 599–608, 2000.
32. N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *In Proc. of the 35th IEEE FOCS*, pages 577–591, 1994.
33. C. Neidle, S. Sclaroff, and V. Athitsos. Signstream: A tool for linguistic and computer vision research on visual-gestural language data. *Behavior Research Methods, Instruments and Computers*, (to appear).
34. S. Park, W. Chu, J. Yoon, and C. Hsu. Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases. *In Proceedings of ICDE*, pages 23–32, 2000.
35. W. Park, X. Zhang, C.B. Woolley, J. Foulke, U. Raschke, and D.B. Chaffin. Integration of magnetic and optical motion tracking devices for capturing human motion data. In *Proc. SAE Human Modeling for Design and Engineering Conference*, 1999.
36. S. Perng, H. Wang, S. Zhang, and D. S. Parker. Landmarks: A New Model for Similarity-based Pattern Querying in Time Series Databases. *In Proceedings of ICDE*, pages 33–42, 2000.
37. D. Rafiei and A. Mendelzon. Querying Time Series Data Based on Similarity. *IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No 5.*, pages 675–693, 2000.
38. D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *Proceedings of Third European Conference on Computer Vision*, 1996.
39. R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *Proc. IEEE Workshop on the Interpretation of Visual Motion*, June 1998.
40. R. Rosales and S. Sclaroff. Specialized mappings and the estimation of body pose from a single image. In *IEEE Human Motion Workshop. Austin, TX*, 2000.
41. H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-26(1):43–49, February 1978.
42. H.W. Sorenson. Least-Squares Estimation: From Gauss to Kalman. *IEEE Spectrum*, Vol. 7, pp. 63-68, 1970.
43. C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.
44. R. Szeliski and S. Bing Kang. Recovering 3D shape and motion from image streams using non-linear least squares. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1993.
45. J. Uhlmann. Satisfying general proximity / similarity queries with metric trees. *IPL: Information Processing Letters*, 40:175–179, 1991.
46. X. Wang and A.R. Hanson. Parking lot analysis/ visualization using multiple aerial images. In *Workshop on Applications of Computer Vision*, page Session 1B, 1998.
47. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real time tracking of the human body. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):780-785, 1997.
48. B.-K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. *In Proceedings of VLDB, Cairo Egypt*, September 2000.