Boston University Computer Science Tech. Report No. 2004-017, April 23, 2004.
To appear in *Proc. International Conf. on Pattern Recognition*, Cambridge, UK, August 2004.

Extraction and Clustering of Motion Trajectories in Video

Dan Buzan
Boston University
dbuzan@cs.bu.edu

Stan Sclaroff
Boston University
sclaroff@cs.bu.edu

George Kollios
Boston University
gkollios@cs.bu.edu

Abstract

A system is described that tracks moving objects in a video dataset so as to extract a representation of the objects' 3D trajectories. The system then finds hierarchical clusters of similar trajectories in the video dataset. Objects' motion trajectories are extracted via an EKF formulation that provides each object's 3D trajectory up to a constant factor. To increase accuracy when occlusions occur, multiple tracking hypotheses are followed. For trajectory-based clustering and retrieval, a modified version of edit distance, called longest common subsequence is employed. Similarities are computed between projections of trajectories on coordinate axes. Trajectories are grouped based, using an agglomerative clustering algorithm. To check the validity of the approach, experiments using real data were performed.

1. Introduction

This paper describes an approach for a computer vision problem: tracking multiple objects in video, estimating their 3D trajectories, and finding groups of similar motion trajectories via hierarchical clustering. The problem arises in a number of important applications, including surveillance and monitoring, remote sensing, transportation safety and traffic analysis systems, shopper flow analysis in stores, etc. Given the importance of the potential applications, there has been significant research activity directed towards solving problems related to tracking motion of multiple moving objects in video.

A number of systems have been proposed for tracking moving objects and estimating their trajectories. For instance [1] use a Gaussian mixture model for modeling the background and detecting potential moving objects. They track moving blobs in consecutive video frames via a linearly predictive multiple hypothesis framework. The moving objects (cars, people) in the video sequence are tracked using Kalman filters. A different Kalman filter-based approach was employed in [4], where an active shape model modeled the contour of a person in each video frame. In [15] detection of moving objects is done using optical flow, and graph representation of moving objects is employed. In [6] a real time surveillance system is proposed that employs a second order motion model and the matching strategy consists of two stages: estimation of object displacement and binary edge correlation between the current and previous silhouette edge profiles. In the abovementioned systems, once trajectories are estimated then time-series indexing and clustering methods can be employed to identify similar motions present in the videos. However, dimensionality reduction is a key issue.

In the data mining community, [19] proposed a method for indexing time sequences that is based on Discrete Fourier Transform (DFT). This work assumed that only a few frequencies are important, and the DFT preserves the Euclidean distance in the frequency domain. In a different approach, FastMap [18] reduces the dimensionality via projection onto k mutually orthogonal directions such that the relative distances between original objects will remain the same for their projections in the k-d space. An improved version [21] computes the similarity of time sequences via Dynamic Time Warping, and FastMap is used to select time sequences that are close enough to the query argument.

In a different approach [17] the time series similarity measure allows one of the two sequences to be scaled by any suitable amount and translated adequately in finding any matching subsequences. Two subsequences are similar if one lies within an envelope of width ε around the other one, ignoring outliers. In [16, 20] similarity of sequences is defined using the Longest Common Subsequence (LCSS) [5]. The measure is expressed as a triple (F, γ, ε) where F is a set of transformation functions, γ is a parameter that controls the ratio of sequences length and ε controls the size of the interval where the mapped value should be.

In this paper, the LCSS approach will be used for grouping similar motion trajectories in an agglomerative clustering algorithm. This allows discovery and retrieval of similar motion trajectories in a video collection. To support this task, the complete system includes video analysis modules: background modeling, moving blob detection, motion prediction and occlusion handling via a Kalman filter, and 3D trajectory estimation.

2. Estimation and Prediction of Trajectories

Objects and their trajectories must be extracted reliably from video sequences. We will need to find a model that is able to emulate the motion of an object in 3D space, which is robust enough to withstand to various influences exerted by the environment. Also, our model will have to take into account the noise injected by the capturing device. The approach presented here is based in large part on [7], with two extensions: 1.) occlusions are handled via multiple hypothesis tracking, 2.) under certain conditions we employ a ground-plane constraint in the estimation of 3D motion trajectories.

The first step consists of initializing the system by collecting enough video frames for computing a statistical background model (mean and covariance of each pixel). After initialization, moving objects are segmented using maximum likelihood estimation. Next, binary image morphology and connected components analysis is employed to convert the foreground pixels that we got after segmentation into entities, blobs, $b_i(t)$, where t is the index of the current frame.

The blobs that we get at this point represent the moving objects in the scene. Nevertheless, the results that we obtain may contain inaccurate information because of occlusions or similarity between the background and objects. In order to gain reliable estimates of the blob motion trajectories, we employ an Extended Kalman Filter (EKF). The trajectories associated with the moving objects are built step by step by matching blobs in the previous frame with the blobs in the current frame. For every frame we match each blob $b_i(t)$ of frame t with zero, one or more blobs $b_l(t\text{-}1)$ of frame $t\text{-}1$. In order to be able to estimate and predict the blob trajectory, we assign to each blob a tracker unit T_j. A tracker unit is designed using the EKF described in [7]. The goal is that the tracker associated with each object will keep this association until the object disappears from the scene. The tracker's role is important when blobs are occluded. In this case the information stored in a tracker unit will help us decide what hypothesis to follow. The trajectories will represent the output of this part of our system.

3. Trajectory Similarity and Clustering

Our goal is to define a distance between trajectories that will take into account the following factors:

a. Different sampling rates (for the cameras) and different speeds (for the objects)
b. Similar motions in different space regions
c. Outliers (noise created by capturing devices)
d. Time sequences with different lengths
e. Computational efficiency

The Longest Common Subsequence (LCSS) algorithm [5] meets these requirements (except b). The algorithm presented here will extend LCSS in 3D (motion will be in a plane, while the third dimension will be time) and will handle the requirement (b) as well.

The approach presented here is based on [16], but is different in that we compute the distance between two time sequences as a pair of numbers. Each number will represent the similarity between the projections of the two time sequences on the coordinate axis. This reduces the computational complexity from cubic, to quadratic in the number of samples for a time series. Also, in defining the LCSS we use a threshold to limit the ratio between the lengths of the two time sequences.

3.1 Similarity Measures

Let A and B two data sequences with sizes n and m respectively. $A = ((a_{x,1}, a_{y,1}), \ldots, (a_{x,n}, a_{y,n}))$ and $B = ((b_{x,1}, b_{y,1}), \ldots, (b_{x,m}, b_{y,m}))$. The projection of A on the x-axis will be denoted as $A_x = (a_{x,1}, \ldots, a_{x,n})$. Also, we will use the functions $Head(A)$ defined as $Head(A) = ((a_{x,1}, a_{y,1}), \ldots, (a_{x,n-1}, a_{y,n-1}))$ and $Head(A_x) = (a_{x,1}, \ldots, a_{x,n-1})$.

Definition 3.1 Given an integer δ and real numbers $0 < \varepsilon$ and $0 < \rho \leq 1$, we define $LCSS_{2D}(\delta, \varepsilon, \rho, A, B)$ as follows:

$$LCSS_{2D}(\delta,\varepsilon,\rho,A,B)=\begin{cases}(0,0) & \\ & if\ \min\{m,n\}<\rho\cdot\max\{m,n\} \\ (LCSS_{\delta,\varepsilon}(A_x,B_x),LCSS_{\delta,\varepsilon}(A_y,B_y)) & \\ & otherwise\end{cases} \quad (3.1)$$

where ρ is a constant Length Aspect Ratio (LAR) that controls the difference in size between the length of the shorter sequence and the length of the longer sequence. The $LCSS_{\delta,\varepsilon}(A_x, B_x)$ is then defined as follows:

$$\begin{cases}0, & \\ & if\ A\ or\ B\ is\ empty \\ 1+LCSS_{\delta,\varepsilon}(Head(A_x),Head(B_x)), & \\ & if\ |a_{x,n}-b_{x,m}|<\varepsilon\ and\ |n-m|\leq\delta \\ \max(LCSS_{\delta,\varepsilon}(Head(A_x),B_x),LCSS_{\delta,\varepsilon}(A_x,Head(B_x))), & \\ & otherwise\end{cases} \quad (3.2)$$

where δ is a constant that controls how far we can look in the past and ε is a constant that controls the size of proximity in which we are looking for matches.

Definition 3.2: The similarity $S1_{2D}(\delta, \varepsilon, A, B)$ between two trajectories A and B, given δ and ε is as follows:

$$S1_{2D}(\delta,\varepsilon,A,B)=\left(\frac{LCSS_{\delta,\varepsilon}(A_x,B_x)}{\min(n,m)},\frac{LCSS_{\delta,\varepsilon}(A_y,B_y)}{\min(n,m)}\right). \quad (3.3)$$

Another element that we will need in defining the new measure is the family of translations F. A translation is represented as a couple (c_x, c_y), where the components are the values of displacement in each dimension:

$$F=\left\{f_{c_x,c_y}\mid f_{c_x,c_y}(A)=((a_{x,1}+c_x,a_{y,1}+c_y),\ldots,(a_{x,n}+c_x,a_{y,n}+c_y))\right\}$$

Definition 3.3: Given δ, ε and the family F of translations we define the similarity function $S2_{2D}(\delta, \varepsilon, A, B)$ between two trajectories A and B as follows:

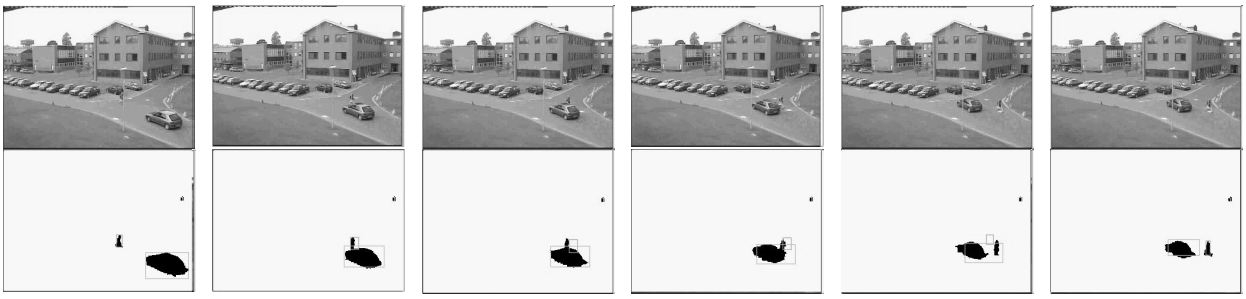

Fig 1. Tracking of multiple objects: before during and after occlusion

$$S2_{2D}(\delta,\varepsilon,A,B) = \max_{f_{c_x,c_y}\in F} S1_{2D}\left(\delta,\varepsilon,A,f_{c_x,c_y}(B)\right)$$

The similarity measure $S2_{2D}(\delta, \varepsilon, A, B)$ is an enhancement of $S1_{2D}(\delta, \varepsilon, A, B)$ because can compute the similarity between sequences that are in different space regions.

Definition 3.4: Given δ and ε we define the distance function between two trajectories A and B:

$$D2_{2D}(\delta,\varepsilon,A,B) = \frac{1}{S2_{2D}(\delta,\varepsilon,A,B)}$$

Note that $S2_{2D}(\delta, \varepsilon, A, B) \in [0,1]$. Therefore $D2_{2D}(\delta, \varepsilon, A, B) \in [1,\infty]$. Also $D2_{2D}(\delta, \varepsilon, A, B)$ is a symmetric function because $LCSS_{2D}(\delta, \varepsilon, A, B) = LCSS_{2D}(\delta, \varepsilon, B, A)$ and translation is a transformation that preserves symmetry.

3.2 Clustering

Clustering trajectories is the last step of trajectory processing. For this we use a modified version of the agglomerative hierarchical clustering algorithm [12]. We chose this algorithm because the data that we get from previous stage is a set of 2D vectors representing the distance between each pair of trajectories. In addition, the number of clusters is unknown. Our method addresses this by computing a dendrogram for the dataset and analyzing the results at the end of the computation.

An important part of the clustering algorithm is the way the distance between two clusters is computed. In our approach we tested two types of distance definitions:

$$d_{\max}(D_i, D_j) = \max_{A\in D_i,\, B\in D_j} \left\| D2_{2D}(\delta,\varepsilon,A,B) \right\| \quad (3.4)$$

$$d_{avg}(D_i, D_j) = \frac{1}{n_i \cdot n_j} \cdot \sum_{A\in D_i} \sum_{B\in D_j} \left\| D2_{2D}(\delta,\varepsilon,A,B) \right\| \quad (3.5)$$

The $d_{max}(\cdot, \cdot)$ is sensitive to outliers. A compromise for improving this problem is the second distance $d_{avg}(\cdot, \cdot)$. Moreover, we can use $d_{avg}(\cdot, \cdot)$ in any algorithm where we used the other distance and complexity of computing $d_{avg}(\cdot, \cdot)$ is similar to that of computing $d_{max}(\cdot, \cdot)$.

The inter-trajectories distance measure is not a metric. So we instead require the clusters to meet two criteria: 1) any two trajectories that have infinite distance between them will be placed in different clusters, and 2) distances between sequences within a cluster will be smaller than the distance to outside trajectories.

4. Experiments

The system was implemented and tested on a dual-processor 1 GHz PC with 1 GB RAM. We tested our system using real and synthetic data. The example video sequence presented here is taken from [14].

Fig. 1 shows six frames from the video clip together with the extracted blobs. Frames 2, 3 and 4 show tracker behavior during an occlusion. Both trackers manage to follow their object, despite the occlusion. Fig. 2 shows extracted trajectories for the entire sequence.

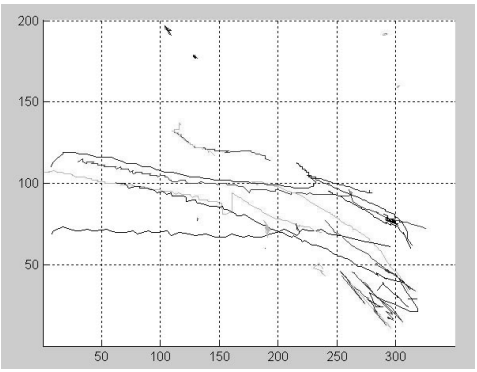

Fig 2. Extracted Trajectories

Using these trajectories, we computed their pair-wise similarities and then clustered them. In computing the similarities we choose ρ = 0.4 (we want to avoid comparison of real trajectories with false trajectories created by noise, which in general are short) and δ = 0.5 (we can go back in time as much as half of the length of the shorter trajectory). For the last parameter, ε, we tested our algorithm using three values 3, 10 and 40. Setting ε = 10 worked best. Finally, the clustering algorithm, described in Section 3.2 was tested using maximum distance (Eq. 3.4) and average distance (Eq. 3.5).

For this set of trajectories, absolute ground truth is unknown. Nonetheless, we can determine minimum number of clusters expected by analyzing initial set of distances between our trajectories subject to the distance criterion (for our example this number will be 8). Upon visually inspecting the trajectories assigned to each cluster, we found that the groupings produced by the

algorithm were intuitively meaningful. For instance, one cluster contained vehicle trajectories along the roadway in the foreground from left-to-right, another cluster contained trajectories in the right-to-left direction, another cluster was noise, etc.

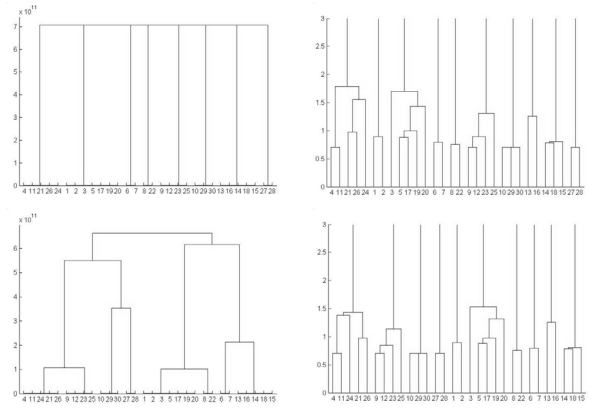

Fig 3. Clusters resulted when ε = 3 (first row–maximum distance, second row–average distance; left column the complete cluster, right column the cluster scaled).

In other experiments [22] with synthetic datasets for which ground truth is available, we found that both distances create the same clusters for a given value of ε. Also, components of a cluster may change when ε or δ changes. Nonetheless, if similarity between trajectories is high than those trajectories will be grouped in the same cluster. Due to space constraints, readers are directed to [22] for a detailed discussion of these experiments.

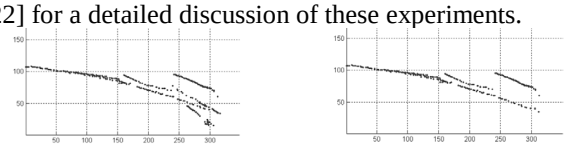

Fig4. Cluster variation when ε changes: left ε=3, right ε=10

Finally, it is noted that computing the distances between the projections of trajectories might have a drawback. Depending on the value of ε we can get a high similarity in one dimension and a low similarity for the other one. In such cases, this can be addressed by decreasing value of ε. Another possibility would be to incorporate a weighted or Mahalanobis distance measure that is tuned via training.

5. Conclusions

We presented a system for estimating trajectories of moving objects from video, for computing the similarities between these trajectories, and then clustering them. Our system can accurately track multiple objects, even in the case when occlusions occur.

We described an extended LCSS formulation for computing the similarity between two time sequences. Having computed the similarities for each pair of trajectories existing in a given video dataset we applied a clustering algorithm for grouping together the trajectories that have common features. For the distance between clusters, we tested two types of distances: maximum distance (Eq. 3.4) and average distance (Eq. 3.5). The generated clusters obey the initial restrictions.

References

[1] C. Stauffer, W.E.L. Grimson, R. Romano, L. Lee. "Using Adaptive Tracking to Classify and Monitor Activities in a Site", *Proc. CVPR*, 22-29, 1998

[2] K. Sato, J.K. Aggarwal. "Tracking Persons and Vehicles in Outdoor Image Sequences Using Temporal Spatio-Velocity Transform", *Proc PETS* 2001

[3] Q. Zhou, J.K. Aggarwal. Tracking and Classifying Moving Objects from Video", *Proc PETS* 2001

[4] N.T. Siebel, S.J. Maybank. "Real-Time Tracking of Pedestrians and Vehicles", *Proc PETS* 2001

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest. "Introduction to Algorithms", MIT Press, 1990

[6] I. Haritaoglu, D. Harwood, L. S. David. "W^4: Who? When? Where? What? A Real Time System for Detecting and Tracking People", *Proc FGRC*, 1998

[7] R. Rosales, S. Sclaroff. "Trajectory Guided Tracking and Recognition of Actions" TR-CS-1999-002, Boston Univ.

[8] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland. "Pfinder: Real – Time Tracking of the Human Body", *PAMI 19(7):* 780-785, 1997.

[9] T. Jebara, A. Azarbayejani, A. Pentland. "3D Structure from 2D Motion", *IEEE Signal Processing Mag.*, 16(3), 1999

[10] M. Koler. "Using the Kalman Filter to track Human Interactive Motion" TR No. 629, Universität Dortmund, 1997.

[11] R. Hartley, A. Zisserman. "Multiple View Geometry in Computer Vision" Cambridge University Press, 2000

[12] R. Duda, P.E. Hart, D.G. Stork. "Pattern Classification" Wiley-Interscience, 2001

[13] R.G. Brown, P.Y.C. Hwang. "Introduction to Random Signals and Applied Kalman Filtering", John Wiley, 1997

[14] Performance Evaluation of Tracking and Surveillance Workshop, PETS 2001, http://pets2001.visualsurveillance.org

[15] I Cohen, G. Medioni, F Bremond, S Hongeng, R Nevatia. "Event Detection and Analysis from Video Streams", *PAMI 23(8)*, 2001

[16] M. Vlachos, G. Kollios, D. Gunopulos. "Discovering Similar Multidimensional Trajectories", *ICDE*, 673-684, 2002

[17] R. Agrawal, K. Lin, H.S. Sawhney, K. Shim. "Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases", *Proc VLDB*, 1995.

[18] C. Faloutsos, K. Lin. "*FastMap*: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets", *Proc ACM SIGMOD*, 24(2), 1995

[19] R. Agrawal, C. Faloutsos, A. Swami. "Efficient Similarity Search in Sequence Databases", *Proc FODO*, 1993

[20] N. Yazdani, M. Özsoyoğlu. "Sequence Matching of Images", *Proc. SSDBM*, 53-63, 1996

[21] B. Yi, H.V. Jagadish, C. Faloutsos. "Efficient Retrieval of Similar Time Sequences Under Time Warping", *Proc ICDE*, 201-208, 1998

[22] D. Buzan, "Robust Tracking of Human Motion" CS-TR-2004-016, Boston University