Dynamics based Trajectory Segmentation for UAV videos

Prithviraj Banerjee Ram Nevatia
Institute for Robotics and Intelligent Systems
University of Southern California, Los Angeles CA 90089
{pbanerje|nevatia}@usc.edu

Abstract

A novel representation of vehicle trajectories is proposed for applications in trajectory analysis and activity detection. Specifically, a piecewise arc fitting based smoothing algorithm is proposed for denoising the trajectories. A dynamic program is used to find the optimal arc fit to a given trajectory. We motivate the usage of dynamic primitives to parametrize common vehicular activities, and propose a dynamics based trajectory segmentation algorithm. Each primitive is modeled using a second order Auto-Regressive model, and form useful descriptors for a given vehicular trajectory. We evaluate both our trajectory smoothing and dynamic trajectory segmentation algorithm on a real UAV video dataset, and show performance improvements which clearly motivate its wide applicability in a general trajectory analysis system.

1. Introduction

Videos from an Unmanned Aerial Vehicle (UAV) platform have become ubiquitous in the surveillance community. As the number of video samples are increasing, automated analysis of these videos have become imperative. In recent publications [10, 16, 1, 9, 11], there has been a lot of focus on analyzing the trajectories of humans and vehicles to recognize the ongoing activity. A common approach is to estimate the homography transformation from each video frame to the world coordinate system, and transform the detection results in each video frame to a trajectory in the world coordinate frame. The stabilized trajectory is used as input to the modeling and classification algorithms.

Unfortunately the raw trajectories obtained after stabilization have considerable amount of noise present in them. The noise originates from inaccurate detections and also imprecise homography estimation. Actual trajectories of vehicles are expected to be relatively stable with a smooth motion profile. Trajectories of common vehicular actions like U-turns, three point turns, parking etc, can be assumed to be constrained to the set of simple parametric curves. This motivates our assumption that the trajectory of a vehicle can be truthfully modeled using simple planar arcs, namely the line-segment and circular arc.

Dynamics of an object refers to the motion produced in the object induced by external forces. In the context of vehicles, dynamics constitute motions like acceleration, braking or making a turn on the road. The entire vehicle trajectory can be segmented into such *dynamic primitives*. Such a representation can be compared to word level granularity, where sentences can represent the action being undertaken. The resulting segments can be used by an appropriate clustering or classification algorithm for subsequent trajectory analysis.

We propose a dynamics based trajectory representation, which can act as a more convenient and intuitive starting point for most trajectory and activity analysis systems. We introduce a novel piecewise arc fitting based trajectory smoothing algorithm to preprocess a trajectory corrupted by noise. We also present an Auto Regressive (AR) model based trajectory segmentation algorithm, which divides the trajectory into dynamic primitives. Each dynamic primitive is represented using a single dynamical equation (i.e. a single second order AR model). We show convincing results on trajectory from a real life UAV dataset. Figure 1 show some of the representative frames to be expected from UAV videos.

(a) (b)

Figure 1. Sample frames from a UAV Video. Note the low resolution makes accurate tracking difficult.

The paper is organized as follows: Sec.2 discusses the existing approaches in the literature. Sec.3 gives an overview of our system. Sec.4 presents our trajectory smoothing algorithm. Sec.5 presents our algorithm for trajectory segmentation and finally Sec.6 presents our results and discussions.

2. Related Work

Morris et al [8] provide a comprehensive survey on video based trajectory analysis for surveillance. They describe the main steps in a trajectory analysis as Preprocessing, Clustering and Modeling. Our paper compliments the methods described in [8], as we propose a new preprocessing step and a suitable trajectory representation in terms of constant dynamics which would greatly improve any subsequent clustering and modeling algorithm.

Numerous methods have been suggested for trajectory smoothing in the computer vision community, of which most of them involve using a simple low pass filter. [6] uses a moving average filter as a preprocessing step before trajectory clustering. [7] uses low-resolution signal decomposition such as wavelets for smoothing. For recognizing vehicular activities from trajectories, the trajectories can be treated as simple 2-D planar curves. A comprehensive review of curve fitting algorithms is given in Horng et al [5]. Our piecewise arc fitting algorithm is closely related to the work by [5]. Their emphasis is on finding optimal approximations to digital planar curves based on an objective measure of perceptual error, whereas our emphasis is on smoothing noisy trajectories.

Sznaier et al [12] nicely summarize the important role of dynamics in computer vision and image processing. Time series models like AR and ARMA models have been used extensively for this purpose. In computer vision community, Bissacco et al [2] introduced dynamical models for recognizing different types of human gait. The gait dynamics are modeled using ARMA models and compared using a distance function introduced by [14]. Veeraraghavan et al [15] have used Autoregressive models for matching shape sequences in Videos. Turaga et al [13] provide a method to recognize the the activity boundaries in a video using a cascade of dynamical models (ARMA models). Ding et al [4] use Hankel matrices to segment a trajectory based on where the dynamics change, though at a much coarser granularity when compared to our method.

3. Overview

Problem Definition: Given a temporally ordered set of tuples $T = \{p_i = (x_i, y_i)\}_{i=1}^{N}$ representing the trajectory of a mobile object from an UAV video, our task is to determine the segments $S_{ij} = \{p_r\}_{r=i}^{j}$ such that each segment can be described by a single dynamical equation (in this case a single second order Auto Regressive matrix). We assume that the trajectory is given in the global coordinate frame. As an illustrative example, Figure 2 shows the trajectory T with red points connected by blue lines. $S_{1,6}, S_{7,16}, S_{17,23}$ are the set of constant dynamic segments determined by the algorithm.

Our solution primarily consists of two stages: Firstly we apply our piecewise arc fitting based smoothing algorithm to T (described in Sec.4), and obtain a smoothed trajectory. We next apply a dynamic programming algorithm to determine the set of segments $\{S_{ij}\} \subset T$ such that each segment S_{ij} represents a constant dynamics of the vehicle (described in Sec.5. Our goal is similar to [13] in the sense that we also aim to identify the *dynamics-change* boundaries in the vehicle trajectory.

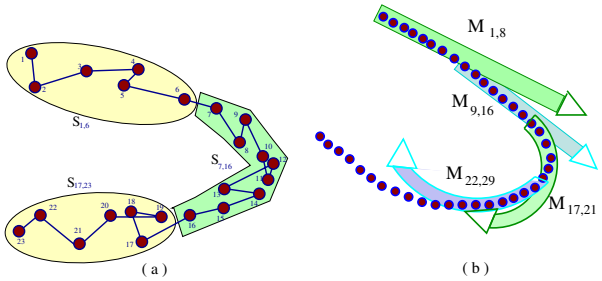

Figure 2. (a) Segmentation of a trajectory into *Dynamics Segments*. (b) Estimated AR Models for given Trajectory

4. Trajectory Smoothing

We fit a piecewise arc consisting of a sequence of line segments and circular arcs to fit the points $\{p_k\}$ on the trajectory. Consider Figure 3(a) as an illustrative example. The red points linked by blue lines represent a sample homography stabilized trajectory. The green segments, show the line segments and circular arc fitted to the trajectory. The number of arc segments to be used and their endpoints are determined using dynamic programming. We obtain a smoothed trajectory by projecting the original trajectory points onto the arc segments as shown in Figure 3(b).

The piecewise arc fitting algorithm is based on the work by Horng et al [5]. We have introduced changes to the algorithm to make it more amenable for fitting to noisy trajectories. Firstly, we use the Douglas Peucker algorithm to find an initial approximate down sampling of the original trajectory, which significantly increases the speed of the subsequent dynamic programming algorithm. Secondly, we identify multiple degenerate cases in the arc fitting algorithm which have not been addressed in [5]. Failure to handle these cases can result in the algorithm *hallucinating* parts of the trajectory without any actual points to support them.

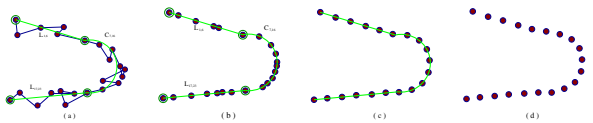

Figure 3. Piecewise arc fitting based trajectory smoothing

4.1. Piecewise Approximation of a Trajectory

In the following sections we describe how to fit a line and a circular arc to a given set of points. We next use a dynamic programming algorithm to determine the optimal sequence of arcs to approximate the trajectory. Let $T_{ij} = \{p_k\}_{k=i}^{j}$ be a temporally ordered set of tuples containing the x and y coordinate of the points on a curve.

4.1.1 Fitting a Line

A straight line $L_{ij} : Y = mX + c$ is fit to the set of points in T_{ij} by choosing the line passing through the first point and the last point in T_{ij}. The equation of the line L_{ij} passing through $p_i = [x_i, y_i]$ and $p_j = [x_j, y_j]$, defined by the constants m and c is given as:

$$m = \frac{y_j - y_i}{x_j - x_i}, \qquad c = y_j - mx_j \tag{1}$$

The error $\varepsilon_L\left(T_{ij}, L_{ij}\right)$ of fitting line L_{ij} to the set of points S_{ij} is given as

$$\varepsilon_L^{ij} = \varepsilon_L\left(T_{ij}, L_{ij}\right) = \sum_{k=i}^{j} d^2\left(p_k, L_{ij}\right) \tag{2}$$

$$d\left(p_k, L_{ij}\right) = \frac{\left(x_k - x_i\right)\left(y_j - y_i\right) - \left(x_j - x_i\right)\left(y_k - y_i\right)}{\left[\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2\right]^{\frac{1}{2}}} \tag{3}$$

where $d\left(p_k, L_{ij}\right)$ is the distance from point p_k to line L_{ij}.

4.1.2 Fitting a Circular Arc

The problem of fitting an arc of a circle $C_{ij}(p_c, r)$ to the set of points T_{ij} does not have a closed form solution. An approximate solution according to [5] is given as follows:

$$p_c = [x_c, y_c] = \left[-\frac{\sum_{k=i}^{j} K_1 K_2}{\sum_{k=i}^{j} K_1 K_3} \quad , \quad ax_c + b\right] \tag{4}$$

where the constants are defined as follows:

$$a = -\frac{x_j - x_i}{y_j - y_i}, \quad b = \frac{y_i + y_j}{2} - a\frac{x_i + x_j}{2} \tag{5}$$

$$K_1 = -x_i - ay_i + x_k + ay_k$$

$$K_2 = x_i^2 + (y_i - b)^2 - x_k^2 - (y_k - b)^2$$

$$K_3 = -2x_i - 2a(y_i - b) + 2x_k + 2a(y_k - b)$$

$$r = \left[\left(x_i - x_c\right)^2 + \left(y_i - y_c\right)^2\right]^{\frac{1}{2}} \tag{6}$$

The points $p_i = [x_i, y_i]$ and $p_j = [x_j, y_j]$ divide the circle C_{ij} into two arcs : major and minor. To determine which arc actually represents the set of points T_{ij}, we compute the set of angles $\Phi = \{\phi_k\}_{k=i}^{j} : \phi_k = angle(\overrightarrow{q_i q_k}, \overrightarrow{q_k q_j})$ where q_k is the projection of point p_k onto C_{ij} (Section 4.2). From general properties of circles, T_{ij} lies on the major arc if $\forall \phi_k : \phi_k < \frac{\pi}{2}$, otherwise it lies on the minor arc. The error $\varepsilon_A\left(T_{ij}, C_{ij}\right)$ of fitting arc C_{ij} to the set of points T_{ij} is given as:

$$\varepsilon_A^{ij} = \varepsilon_A\left(T_{ij}, C_{ij}\right) = \sum_{k=i}^{j} d^2\left(p_k, C_{ij}\right) \tag{7}$$

$$d\left(p_k, C_{ij}\right) = r - \left[\left(x_k - x_c\right)^2 + \left(y_k - y_c\right)^2\right]^{\frac{1}{2}} \tag{8}$$

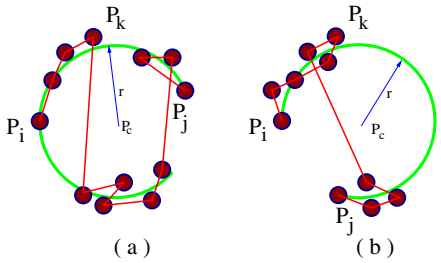

Figure 4. Degenerate cases in circular arc fitting

Handling degenerate cases : The arc fitting algorithm can result in some undesirable fits to the given set of points. We identified two cases that arise due to a degenerate arrangement of the points on a circle. Consider the case depicted in Figure 4(a), where the arc-fitting algorithm will return a valid fit, but clearly an undesirable fit. In general, we do not want to fit a single arc of a circle to such arrangements of data points. To avoid these cases we need to ensure that all the angles $\Phi = \{\phi_k\}$ are either greater than $\frac{\pi}{2}$ or less than $\frac{\pi}{2}$. Otherwise the corresponding error ε_A^{ij} is set to ∞. The second degenerate case results when enough points are not present to justify a circular arc through them, especially when the points are not well spread out over the arc. We need to enforce a constraint which ensures that the points $p_k \in T_{ij}$ are roughly uniformly distributed over the arc. Mathematically this translates to ensuring $\theta_{max} - \theta_{min} < \pi/2$, where

$$\theta_k = angle(\overrightarrow{q_k q_i}, \overrightarrow{q_i q_j})$$

$$\theta_{max} = \max_{k \in [i \ldots j]} \theta_k, \qquad \theta_{min} = \min_{k \in [i \ldots j]} \theta_k$$

For all sets of points T_{ij} unable to satisfy the above constraint, we set $\varepsilon_{Arc}^{ij} = \infty$.

4.1.3 Dynamic Program for optimal fit

Our objective is to fit a set of line segments and circular arc segments to an ordered set of points $T = \{p_k\}_{k=1}^{N}$. The optimization criteria is to reduce the fitting error η_e of the curve, but at the same time avoid fragmentation of the curve into too many small segments. This is achieved by imposing a penalty term η_f for every fragment generated by the algorithm, and hence ensuring it does not trivially fit segments to consecutive pairs of points, resulting in zero fitting error with $N-1$ segments. The Dynamic Programming (DP) algorithm pseudo code closely follows the one in [5] and is given as Algorithm 1.

Algorithm 1 Dynamic Programming Algorithm for Piecewise Arc Fitting

INPUT: η_e, η_f, $T = \{p_k\}_{k=1}^{N}$, where N is total Data Points
PARAM LIST: $\eta_l = 1, \eta_a = 1.5$

Let Optimal path matrix $Opt \in R^{N\times 2}$
Let Error matrix $E \in R^{N\times N}$

Set $Opt_{ij} = 0 \quad \forall i,j$
Set $E_{ij} = 0 \quad \forall i,j$

for $\forall i,j: \quad i \le j$ **do**
$E_{ij} = Min\left\{\eta_l\left(\varepsilon_L^{ij} + \frac{1}{j-i+1}\right), \eta_a\left(\varepsilon_A^{ij} + \frac{1}{j-i+1}\right)\right\}$
end for

for j= 2 to N **do**
Let $Q_{ij} = \eta_e E_{ij} + \eta_f$
$Opt_{j1} = \min_{1\le i\le j}(Q_{ij} + Opt_{i1})$
$Opt_{j2} = \operatorname{argmin}_{i:1\le i\le j}(Q_{ij} + Opt_{i1})$
end for
Backtrack on $OPT_{i2} : i = N \ldots 1$ to obtain optimal Curve Fit.

4.2. Projecting onto Arc Segments

We project the original trajectory points onto the corresponding segments determined using Algorithm 1. To project a point p_k on a given line $L_{ij} : Y = mX + c$, we use the following equation :

$$x_p = \frac{my_k + x_k - mc}{m^2+1}, \qquad y_p = \frac{m^2 y_k + mx_k}{m^2+1} \tag{9}$$

To project a point p_k on a given Circular arc $C_{ij}(p_c, r)$ use the following equation:

$$\theta = \arctan\left(\left|\frac{y_k - y_c}{x_k - x_c}\right|\right) \tag{10}$$

$$x_p = \begin{cases} x_c + r\cos\theta & \text{if } x_k \ge x_c \\ x_c - r\cos\theta & \text{if } x_k < x_c \end{cases} \tag{11}$$

$$y_p = \begin{cases} y_c + r\sin\theta & \text{if } y_k \ge y_c \\ y_c - r\sin\theta & \text{if } y_k < y_c \end{cases} \tag{12}$$

The projected points need to be sorted to maintain the temporal consistency in the ordering of the trajectory points.

4.3. Douglas Peucker Algorithm for speedup

Dynamic programming based algorithms need to compute the Error matrix $E \in R^{N\times N}$, which requires $O(N^2)$ computation. Hence for large trajectories, the running time is quadratic in the number of data points. A particularly bad case is that of a large set of co-linear points, where it is trivial to fit a single straight line to it. The dynamic programming algorithm assumes the worst case scenario, and hence computes the entire E matrix and attempts to find the best set of fit on the points. Such cases are quite common in the real world, as for example a vehicle on a highway is expected to travel in a straight line for longer periods of time, and hence smoothing such regions of the curve should be handled in a trivial manner.

A simple solution might be to sample the trajectory points to reduce the number of points to be considered. This will alleviate our problem in straight line motion, but will cause problems at curvature points. For example, consider the trajectory in Figure 5(a). Severe under sampling causes the entire turn to disappear as seen in Figure 5(b). Hence the sampling factor needs to be decided for each curve separately, depending on how much detail we want to preserve on the trajectory.

The above comments suggest that we need to sample the trajectory in an intelligent manner, especially we should sample long straight portions at a smaller rate, and high curvature areas at a higher sampling rate. Hence we use the Douglas-Peucker algorithm [3] (also called *iterative end-point fit* method) to get a rough piecewise linear approximation of the trajectory as shown in Figure 5(c). The algorithm uses a greedy selection strategy to recursively divide the trajectory, and approximate each division with a line segment. The recursion continues until the desired level of aproximation is achieved. Hence, the algorithm approximates large collinear sets of points by a single line segment and approximates high curvature regions by a series of shorter line segments. We set the sampling rate inversely proportional to the length of the line segment approximating the curve in that region. Hence we use a larger sampling rate for the high curvature parts of the trajectory, and a smaller sampling rate for the straighter parts of the trajectory. Figure 5(d) shows the sampled trajectory based on the described algorithm. The down-sample trajectory constitutes the input $T = \{p_k\}$ in Algorithm 1, and hence the dynamic program algorithm computes a much smaller E matrix. The

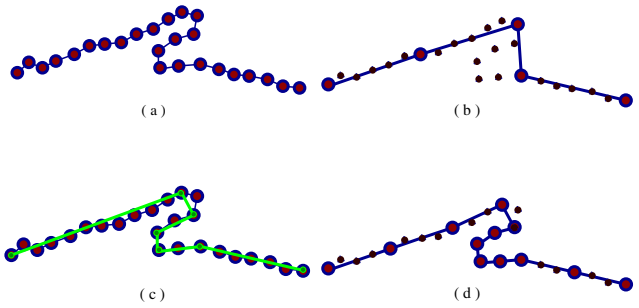

Figure 5. (a) Original trajectory. (b) Severe under sampling (c) Piecewise linear approximation of trajectory. (d) Intelligent sampling

full description of the piecewise arc fitting based trajectory smoothing algorithm is given in Algorithm 2.

Algorithm 2 Piecewise Arc Fitting based Trajectory Smoothing

1: INPUT: $\eta_e, \eta_f, S_{1N} = \{p_k\}_{k=1}^{N}$, where N is total number of trajectory points.
2: Apply Douglas-Peucker algorithm to determine a piecewise linear approximation of the trajectory and down-sample it accordingly.
3: Run the Dynamic Programming (DP) algorithm and obtain a piecewise arc approximation of S_{1N}.
4: Project the points T on to the fitted curves to obtain T_{prj}.
5: Smoothen T_{prj} at the junction points of the piecewise arc.
6: Return T_{prj} as final Output.

5. Dynamics based track segmentation

We consider two cases- first in which we retain the original trajectory points projected on to the piecewise arc segments, and hence preserve the original dynamics of the vehicle. Second, we simplify the problem, and assume that the vehicle always travels at a constant speed (not constant velocity). To enforce the constant speed assumption, we resample the smoothed trajectory at constant intervals (Figure 3c). This is possible as we already have a piecewise parametric model of the smoothed trajectory in form of Line and Arc equations of each fragment. The rationale behind the two cases is as follows: Actions like making left/right turns, U-turns and straight motion are characterized only by the change in direction. Two vehicles executing the same left-turn motion may have completely different dynamical equations, one deaccelerating while making the turn, and the other accelerating. Hence to detect such directional dynamical primitives, it is easier to remove the speed information from the given trajectory.

Our algorithm for detecting the dynamic primitives closely follows Algorithm 1. Instead of detecting piecewise arc fits to a trajectory, we would like to find piecewise AR model fits. In terms of the dynamic program, we need to define an appropriate error matrix $E \in R^{N \times N}$. We take a subset of points $T_{ij} \subset T$ from the trajectory and estimate a k^{th} order (where $k < j - i$) AR model M_{ij}, which is best able to predict the trajectory in T_{ij}. An k^{th} order Auto regressive model is defined as follows:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \sum_{j=1}^{k} A_j \begin{pmatrix} x_{i-j} \\ y_{i-j} \end{pmatrix} + A_0 \tag{13}$$

We use the *MATLAB* function *arx()* to estimate the parameters $\{A_r\}_{r=0}^{k}$ which is a least square estimation problem. The *arx()* function solves the overdetermined set of linear equations using QR factorization. We compute the prediction error as the average error of the AR-Model in predicting l steps ahead, where $l = 3$ in our case. For example, given the points $\{[x_k, y_k]\}_{k=1}^{3}$, we predict the points $\left\{\left[x_k^{'}, y_k^{'}\right]\right\}_{k=4}^{6}$ by recursively applying Equation 13. The prediction error E_{ij} is then given as the euclidean distance between $[x_6, y_6]$ and $\left[x_6^{'}, y_6^{'}\right]$. Hence, we can compute the error matrix E and apply the dynamic program given in Algorithm 1 to find the dynamic primitives $\{S_{ij}, M_{ij}\}$ of the trajectory.

Figure 2(b) shows the estimated AR models of the given trajectory (under constant speed assumption). Each AR model is able to predict accurately for its given segment only. Beyond a certain point, each model diverges from the actual trajectory, which signifies a change in dynamics of the car. Hence, we obtain the segments $S_{ij} = \{p_r\}_{r=i}^{j}$ with the corresponding AR model M_{ij} for the entire trajectory.

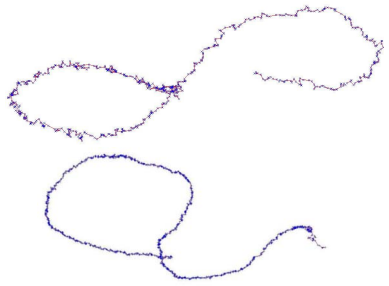

Figure 6. Raw trajectory after homography stabilization

6. Results and Discussions

We perform our experiments on vehicle trajectories obtained from UAV platform videos. UAV videos (Figure 1) generally have poor resolution and high amount of background motion as it is taken from an unstable moving plat-

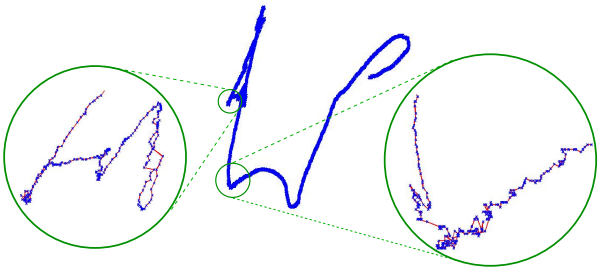

Figure 7. Trajectory examples with magnified segments showing the large amount of noise present.

form. Example trajectories are shown in Figure 6 and 7. Magnification of the trajectory in Figure 7 shows a large amount of noise present due to improper homography stabilization.

We first present results showing the effectiveness of our smoothing algorithm, even in the presence of severe noise. Consider the underlying trajectory fragments (in black) in Figure 8. Both fragments belong to a single trajectory of a vehicle. We first apply a simple moving average filter on both the trajectories. A simple MAF will remove some of the perturbation, but *kinks* will remain on the trajectory which will act as noise for subsequent models. Fixing the window size of the MAF is dependent on trajectory properties like the scale of the turns to be detected, and the level of noise distortion present. Hence the window size needs to be set by manual inspection of the trajectory. Figure 8(a,c) show the smoothed curve in blue with a window size of 10, and Figure 8(b,d) show the results for a window size of 100. Subjectively Figure 8(a,d) constitute the best results for the moving average filter. Our experiment demonstrates that in the general case, the window size needs to be changed dynamically for different parts of the same trajectory. The amount of distortion introduced in Figure 8(b) is simply unacceptable for an activity analysis system. Such distortion might be avoided by reducing the window size, but at the expense of smoothing quality, as observed in Figure 8(c).

We apply Algorithm 2 to the sample trajectories, and the results are given in Figure 9. With the same set of parameters, we achieve qualitatively better results on both segments, and hence clearly demonstrating the usefulness of our proposed algorithm. Some more sample results from the dataset are given in Figure 15.

We next apply the trajectory dynamics segmentation algorithm to the smoothed trajectories, and obtain the segments with constant dynamics. Figure 10 shows the segments identified by the Auto Regressive models under constant speed assumption. We note that with a constant speed, the only changing dynamics is that of the changing direction of motion. Figure 10 shows an example in support of our hypothesis, where boundary points between segments de-

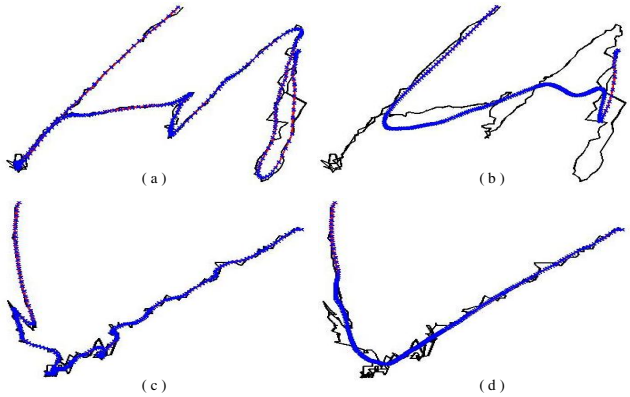

Figure 8. Comparison of Moving Average Filter with different window sizes: (a,c) have window size of 10; (b,d) have window size of 100. Original trajectory is shown in black. The red line with blue markers shows the smoothened trajectory.

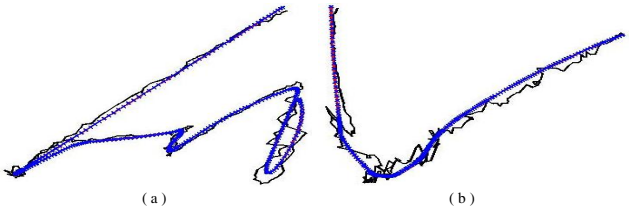

Figure 9. Curve Fitting Based smoothing results

marcate a change in the turning motion of a vehicle. Such a trajectory segmentation provides a powerful input representation for any subsequent activity analysis.

On removing the constant speed assumption, the dynamics are now defined by both changing direction, and changing speed. Figure 11 shows our results in this case. Although this leads to over segmentation of the tracks, but we get a set of discrete segments with a more richer representation, than just individual trajectory points. Each segment describes an acceleration, deacceleration or change in angular momentum of the vehicle. As mentioned before, such a representation can be compared to word level granularity, where sentences can represent the action being undertaken. The boxed region of the trajectory is magnified in Figure 12. We observe that there is clear demarcation in the dynamics of the segments, for example segment (2) corresponds to deacceleration and segment (3) corresponds to acceleration. All segments need not correspond to a clearly definable dynamics, as seen in segment (1). Second order AR models are used to describe the dynamics, which is a much richer representation than just simple acceleration / deacceleration labels.

It is difficult to give an quantitative evaluation of our proposed algorithm, as we cannot easily define a suitable metric for comparing our smoothing and dynamic track seg-

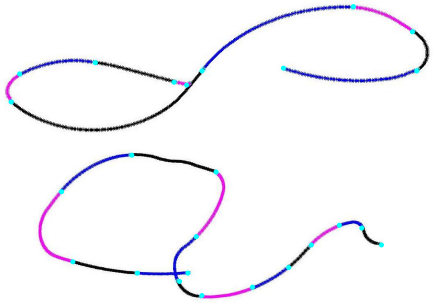

Figure 10. Estimated AR Models for given Trajectory under constant speed assumption. Each color (Black, Blue and Magenta) represents a single AR model segment

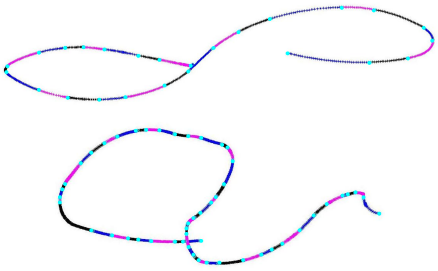

Figure 11. Estimated AR Models for given Trajectory under changing speed assumption. Each color (Black, Blue and Magenta) represents a single AR model segment

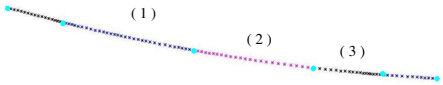

Figure 12. Magnification of the boxed region in Fig.11. Labels (1),(2) and (3) are described in the text.

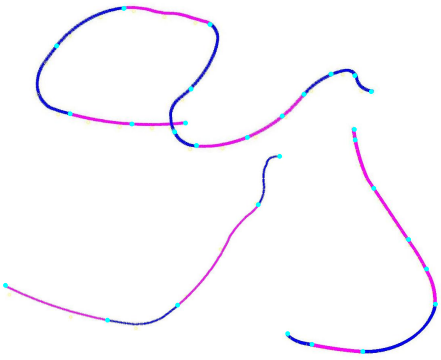

Figure 13. Classification results: Blue is a turn motion, magenta is straight motion.

mentation algorithm to other classical approaches. To validate our hypothesis that the dynamic primitives can indeed be used as input for activity analysis, we perform a simple

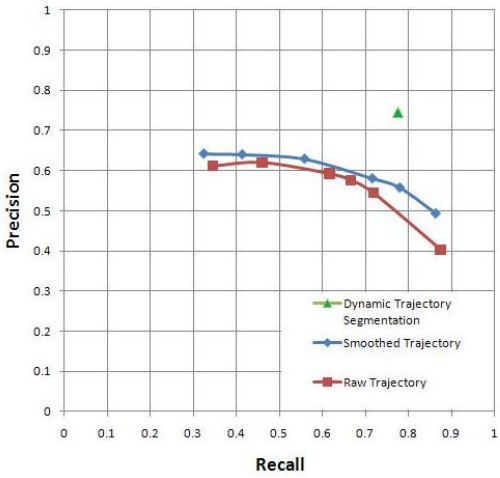

Figure 14. Precision-Recall curve for turn-detection experiment

turn-detection experiment. Our dataset consists of a set of 20 trajectories with annotated ground truth. Our objective is to detect the regions of the trajectory where the vehicle is executing a turn. We consider a simple classifier function $\mathcal{F}(\cdot)$, which takes a set of trajectory points X as input, and classifies the middle point as a turn or a straight motion based on its radius of curvature.

Given the raw trajectory as input, a sliding window approach is used to detect the turn events, by applying the classifier $\mathcal{F}(X_{win})$ to set of points X_{win} contained in the window, and determining the label for the center point. The red curve in Figure 13 shows the PR curve for the raw trajectory case, where each point on the curve corresponds to a different window size and radius of curvature threshold. We next apply the classifier $\mathcal{F}(\cdot)$ to trajectories smoothed by Algorithm 2, and obtain the corresponding blue curve in Figure 13. We see a distinct improvement in the classifier performance by just using our smoothing algorithm.

Lastly, we obtain the dynamic primitives S_{ij} corresponding to the trajectories (under constant speed assumption) using the algorithm described in Section 5. The classifier $\mathcal{F}(S_{ij})$ is applied to each primitive S_{ij}, and classified as a turn or straight motion. We achieve a precision rate of 0.74, and a recall rate of 0.77 (green point in Figure 13), and hence obtain a significant performance improvement even with such a naive classifier for turn-detection. One can easily envision much more complex classifiers and rule based systems which can use these segments and perform trajectory and activity analysis.

7. Conclusions and Future Work

We have presented a novel piecewise arc based trajectory smoothing algorithm, and demonstrated its accuracy on extremely noisy trajectories. In this respect, we propose

Figure 15. Sample results of trajectory smoothing.

the Douglas-Peucker algorithm based down-sampling as a speedup to the piecewise arc fitting algorithm. We identify two degenerate cases in the piecewise arc fitting algorithm which need to be taken care of for proper function of the trajectory smoothing. We also present an Auto-Regressive model based trajectory dynamics segmentation algorithm, which gives a concise representation of the trajectory in terms of dynamic primitives. Such a representation makes it amenable for subsequent activity analysis systems, as showed by the simple turn detection classifier applied by us.

In the future, we would like to explore bag of words based classification approaches utilizing the dynamic primitives detected by our algorithm. We would also like to explore the possibility of augmenting image features along with the trajectory information to help the dynamics segmentation.

Acknowledgment

This material is based upon work supported by the Prime Contractor, BAE Systems under Subcontract #073692 (Prime Contract #HR0011-08-C-0134) issued by DARPA. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of BAE Systems or DARPA.

References

[1] F. I. Bashir, A. A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using hidden Markov models. *IEEE transactions on Image Processing*, 16(7):1912–9, July 2007. 1

[2] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of Human Gaits. *In CVPR*, 51:52–57, 2001. 2

[3] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112–122, 1973. 4

[4] T. Ding, M. Sznaier, and O. Camps. Fast track matching and event detection. In *CVPR*, 2008. 2

[5] J. Horng. A dynamic programming approach for fitting digital planar curves with line segments and circular arcs. *Pattern Recognition Letters*, 22(2):183–197, Feb. 2001. 2, 3, 4

[6] I. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. In *ICPR*, 2004. 2

[7] X. Li, W. Hu, and W. Hu. A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering. In *ICPR*, pages 591—-594, Washington, DC, USA, 2006. 2

[8] B. T. Morris and M. M. Trivedi. A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1114–1127, Aug. 2008. 2

[9] S. Pachoud, E. Maggio, and A. Cavallaro. Grouping motion trajectories. In *ICASSP*, pages 1477–1480, 2009. 1

[10] R. Sillito and R. Fisher. Semi-supervised learning for anomalous trajectory detection. In *BMVC*, 2008. 1

[11] E. Swears, A. Hoogs, and A. A. Perera. Learning Motion Patterns in Surveillance Video using HMM Clustering. In *WMVC*, Jan. 2008. 1

[12] M. Sznaier and O. Camps. The role of dynamics in computer vision and image processing. In *IEEE Conference on Decision and Control*, pages 3923–3934. Ieee, Dec. 2008. 2

[13] P. K. Turaga, A. Veeraraghavan, and R. Chellappa. From Videos to Verbs: Mining Videos for Activities using a Cascade of Dynamical Systems. *In CVPR*, June 2007. 2

[14] P. Van Overschee and B. De Moor. Subspace algorithms for the stochastic identification problem. *Automatica*, 29:649–660, 1993. 2

[15] A. Veeraraghavan, A. K. Roy-Chowdhury, and R. Chellappa. Matching shape sequences in video with applications in human movement analysis. *In PAMI*, 27(12):1896–909, Dec. 2005. 2

[16] X. Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *In ECCV*, pages 110 – 123, 2006. 1