Topological and Statistical Behavior Classifiers for Tracking Applications

Paul Bendich*, Sang Chin†‡§, Jesse Clarke†, Jonathan DeSena†, John Harer *, Elizabeth Munch *, Andrew Newman †, David Porter †, David Rouse †, Nate Strawn *†, Adam Watkins †

* Departments of Mathematics, Computer Science, and Electrical and Computer Engineering, Duke University, Durham, NC 27708

†Johns Hopkins University Applied Physics Laboratory, Laurel, MD, 20723

‡Department of Computer Science, Boston University, Boston, MA, 02215

§Draper Laboratory, Cambridge, MA, 02139

Abstract

This paper introduces a method to integrate target behavior into the multiple hypothesis tracker (MHT) likelihood ratio. In particular, a periodic track appraisal based on behavior is introduced that uses elementary topological data analysis coupled with basic machine learning techniques. The track appraisal adjusts the traditional kinematic data association likelihood (i.e., track score) using an established formulation for classification-aided data association. The proposed method is tested and demonstrated on synthetic vehicular data representing an urban traffic scene generated by the Simulation of Urban Mobility package. The vehicles in the scene exhibit different driving behaviors. The proposed method distinguishes those behaviors and shows improved data association decisions relative to a conventional, kinematic MHT.

Keywords: Multiple Hypothesis Tracking, Topological Data Analysis, Machine Learning, Feature-Aided Tracking

I. Introduction

THERE has been a growing awareness in the tracking and data fusion communities that target behavior provides key information for enabling successful tracking under challenging conditions, particularly when success is based upon Activity-Based Intelligence (ABI). As sensor capabilities and tracking algorithms are employed to achieve real-time, wide area, multi-sensor tracking with confidence, the information that can be provided by target activity and pattern-of-life provide exciting new dimensions to meet these challenges. Furthermore, as tracking methods are extended to the higher-level fusion problems (such as situation awareness and determination of intent) behavior estimation becomes particularly important.

In this paper, we present a method for integrating target tracking using Multiple Hypothesis Tracking (MHT), Topological Data Analysis (TDA), and machine learning. The key idea is to use topologically-inspired measures of behavior to prune out improbable tracklets (contiguous plausible trajectory fragments) inside of the MHT. TDA not only provides classification of target behavior, but (perhaps even more importantly) also helps solve what is often the most difficult problem in target tracking: connecting-the-dots by associating data with targets. Intuitively, data are associated in a manner not only consistent relative to target location-related data and type-related data, but also consistently relative to behavior-related data. For example, tracking urban traffic with Wide Area Motion Imagery (WAMI) data may benefit more from distinguishing vehicles by their behavior than from trying to distinguish vehicles using fuzzy imagery.

This paper uses efficiently-computable topological features to characterize certain agent behaviors. We demonstrate the value of our approach by analyzing a simple, but plausible scenario: disambiguation of agents passing through an intersection. Our test data is generated by the Simulated Urban Mobility (SUMO [15], http://sumo.sourceforge.net) package, which we have modified to simulate tracking data that would be produced by a WAMI sensor system. In addition, we believe that the results presented here only scratch the surface of our potential contributions, as our basic ideas immediately penetrate into areas such as coordinated behavior of groups of targets, behavior of components of extended target systems that may not be collocated (such as an air defense system), and even behavior exploitable by cyber data fusion.

A. Multiple Hypothesis Tracking.

Most modern multi-target multi-sensor tracking systems use some version of the MHT framework pioneered by Reid [21], and comprehensively described in Blackman and Popoli [6] and by Hall and Llinas [16]. For the last decade or two, the dominant approaches are formulated as a Bayesian inference problem that scores competing multi-track hypotheses with a Bayesian Log-Likelihood Ratio (LLR) as described by Bar Shalom and collaborators [2]. Data are processed recursively with a deferred decision logic scheme that expands parent hypotheses with new data to form child hypotheses. These child hypothesis are subsequently pruned to provide a reasonable number of parent hypotheses for the next round of data. The track file of pruned candidate tracklets can become much larger than the actual number of targets, where the tracklets are alternative attempts to represent targets with the same data.

For urban traffic with WAMI data, images from multiple cameras are processed using a motion detection algorithm to segment pixels into candidate vehicles. As vehicles slow, particularly at intersections, motion detection becomes difficult. Further, algorithms for detecting stationary vehicles are much less effective. Consequently, the tracklets can be quite confused at intersections. Often there will be tracklets trying to represent the same vehicle that make all possible turning decisions at the intersection. In other words, it can be difficult to connect-the-dots with fuzzy images of vehicles that are near each other. However, if the behavior for a tracklet after the intersection is similar to behavior before the intersection, then it would seem the score of the tracklet should be increased and that the confusion could be resolved. This is in fact what the behavior flags, coupled with TDA and machine learning accomplish as described below.

The approach taken in this paper is motivated by a previous extension of MHT to include target type classification by Wigren [22] and Bar Shalom, et alia [3]. Many attempts had been made to incorporate target type data in MHT, but most of them were ad hoc. Wigren and Bar Shalom provided a unified approach that used target type data to compute an additive term for the track LLR. Not only was target type classification provided, but data association was improved. In fact, this was an enabling technology for field-scale implementation of Upstream Data Fusion (UDF) methods reported by Newman and Mitzel [19].

Target behavior classification is incorporated into MHT in a similar manner as target type classification. A difference is that target behavior uses a window of data over which the behavior can be observed, whereas target type can be observed based only on current data. Consequently, the target behavior algorithm is multi-rate with the target kinematic state estimation and target type classification running at a faster rate than the target behavior classification. Another point is that both target type and behavior can be incorporated treating classification with current observations as the data, or treating the upstream features used by the classifier as the data.

B. Topological Data Analysis.

To each tracklet we want to associate functions that describe behavior. We focus on speed, acceleration and turning, but the general framework will work for any functions. Figure 1 illustrates speed functions of different behavior types. Given two tracklets T_1 and T_2, we will use these functions to help us figure out whether or not the agents associated to T_1 and T_2 are the same.

There are of course many different ways to summarize functional data: critical values, total variation, motifs, to name a few. Our proposed solution is to use a *persistence diagram* (PD), which we will argue provides a picture of the functional data that is both stable with respect to noise, easy to compute, and captures the important features of each function with no need for horizontal or vertical alignment. We will then use machine-learning methods to classify the persistence diagrams into behavioral types.

The persistence diagram is one of the main tools in Topological Data Analysis (TDA), a new and developing field [11], [13] which adapts methods from algebraic topology to find structure in complex datasets. There have already been many promising applications of TDA to (for example) gene expression [10], signal analysis [20], and orthodontia [14].

PDs provide a robust and low-dimensional picture of some of the multi-scale topological and geometric information carried by a point cloud or a function on some space. A rigorous explanation of the most general type of PD requires a background in algebraic topology. Fortunately, the PDs we use in this paper are from a much simpler context, representing the evolution of connected components for the threshold or sub-level sets of functions $f : [a, b] \to \mathbb{R}$, and can be understood without recourse to advanced techniques in algebraic topology. For a survey on PDs in general, see [12]. The requisite background on algebraic topology can be found in (for example) [18].

Once a PD has been computed, the obvious question is how to interpret it. In most applications to date, the answer has been rather ad-hoc. In this paper, we propose what we hope will be a widely-applied general machine-learning method for interpreting PD datasets in a statistical context. See the related paper [4] for more details on this method.

C. Outline.

The structure of this paper is as follows. In the next section, we describe a MHT tracker that is particularly useful in tracking applications. We then demonstrate that enriching this particular MHT tracker with topological features motivated by behavioral characteristics improves the efficacy of the tracker in our reasonably plausible scenario.

In Section II, we introduce the simple behavior functions associated to each tracklet. An elementary introduction of persistence diagrams, in the context of these functions, comes in Section III. The transformation of these diagrams into features suitable for machine learning is described in Section IV, which also contains the experiments we ran on simulated data to demonstrate that the learned topological features do succeed in picking out behavior types. The technical details behind the incorporation of the topology into the MHT tracker is given in Section V-A, where we demonstrate via example that the new tracker connects-the-dots in a situation where the old tracker remains confused.

II. Behavior Functions

We now introduce a few simple functions that will be used to characterize aspects of driver behavior. The functions used in the work presented here are based on speed, acceleration, and turning radius, but the general framework could certainly incorporate different function types.

A. Functional profiles

Suppose we have a tracklet under consideration which has coordinates

$$\{(x(t_i), y(t_i), z(t_i) : 0 \leq i \leq n\},$$

where n is the number of pixels in the tracklet. Let $v(t_i)$ be the resulting velocity vector, computed for example as a 2-step average:

$$v(t_i) = (\Delta x_i, \Delta y_i, \Delta z_i)$$

with

$$\Delta x_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}},\ \Delta y_i = \frac{y_i - y_{i-1}}{t_i - t_{i-1}},\ \text{and}\ \Delta z_i = \frac{z_i - z_{i-1}}{t_i - t_{i-1}}.$$

From the $v(t_i)$, we discuss how other quantities of interest may be computed.

B. Simple Function Choices

The first function we use is the speed curve of each vehicle:

$$s(t_i) = \|v(t_i)\|.$$

We will also consider the acceleration

$$a(t_i) = \|\dot{v}(t_i)\|$$

estimated similarly to v, and turning, which is a variant of the angular velocity

$$T(t_i) = \frac{\|v(t_i)\| + \|v(t_{i+1})\|}{2} sin(\theta_i)$$

where θ_i is the angle between $v(t_i)$ and $v(t_{i+1})$.

Our hypothesis is that both the speed and the acceleration functions should display strong variation for a more aggressive driver, at least over a reasonably short period of time. The turning function measures how fast agents go when turning. When an agent turns, $\sin(\theta)$ is non-zero, and a faster turn will mean that the norms of the velocity vectors are larger. A car going fast in a straight line won't register, although a "fast lane-switcher" may have small $\sin$ values, but the very large velocity may register the car as a different profile. We look for a large peak for racers, and a smaller peak for slow drivers.

C. A poor choice: transient critical values.

Since we are interested in *practical and computable* invariants of track motion, one might ask why we could not simply use the maximum speed or turning values computed from vehicle velocities. It is easy to see that this is not a particularly good invariant of behavior, and we illustrate this fact with a simple story.

Consider the two speed functions shown in Figure 1. We imagine two vehicles which our camera captures entering and then leaving a freeway with a posted speed limit of 65 miles per hour (MPH). Both vehicles enter the highway, and speed up to $70MPH$. Vehicle A speeds up evenly, maintains mostly constant speed near 70 while on the freeway, and slows down gradually at the exit ramp. Vehicle B accelerates more quickly (guns it), speeds up and slows down during the time on the freeway, perhaps because it passes many cars and tailgates others, and then exits with a sharp deceleration.

The maximum speed may capture very little information to distinguish these two. But if we look at the shape of the speed curves, we see considerably more variation for B. Furthermore, most of this extra variation happens at high speed values. The persistence diagram of the speed function, described in the next section, is ideally suited to summarize all of this information.

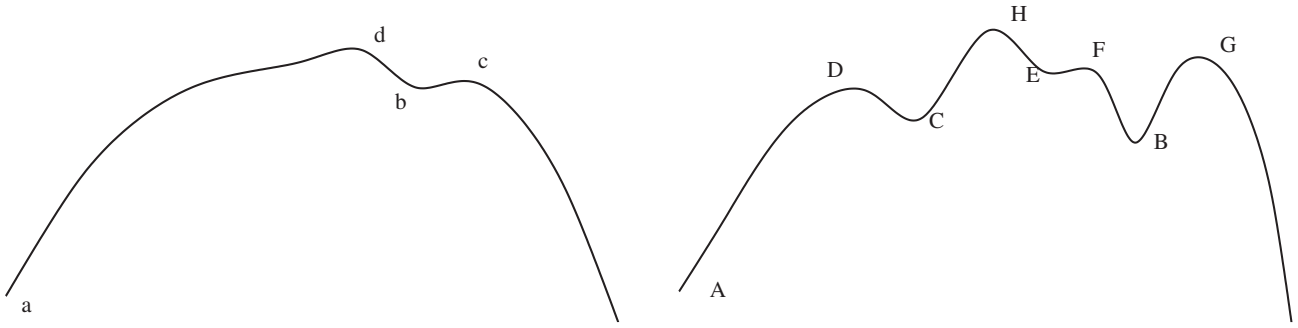


Fig. 1: Possible speed profiles for a non-agressive driver (left), and an aggressive one (right). The letter labels indicate the critical values.

III. Persistence Diagrams

In this section, we introduce the 0-dimensional *persistence diagram* $D_0(f)$, of a function. We will not do so in the most general context, choosing instead to restrict to what is most relevant to this particular application.

A. 0-dimensional persistence for continuous functions

We start with a continuous function $f : [a, b] \to \mathbb{R}$. Typically f is given by its sample values at a set of points $T = \{t_0, \cdots, t_n\} \subset [a, b]$, $a = t_0 < t_1 < \cdots < t_n = b$, and we linearly interpolate to obtain a continuous function. Let $u_i = f(t_i)$, and assume for simplicity that $u_i \neq u_j$ for all $i \neq j$. If this is not the case, a small deformation of values breaks the ties and corrects the situation. Let m and M be the minimum and maximum (respectively) of f on $[a, b]$. By continuity, m and M occur at points in the set T.

Define the sub-level set at level c of f to be

$$f_c = f^{-1}((-\infty, c]) = f^{-1}([m, c])$$

By continuity of the function f, we know that f_c has the form

$$f_c = \bigcup_{k=1}^{K} [a_k, b_k]$$

where each of the closed intervals $[a_k, b_k]$ are disjoint and satisfy $a_k \leq b_k$. Given such a decomposition of f_c, we say that f_c has K *components*, and we track how many components f_c has as the level c goes from m to M. A real number c is called a *critical value* of f if f_c has either more or less components than $f_{c-\epsilon}$ for ϵ sufficiently small[1]. For example, the critical values of the functions in Figure 1 are labeled by letters.

If the number of components goes up when we pass the threshold value c, we say that a *birth* happens at c. In our interpolated case, this happens at a point t_i where both $u_{i-1} > c = u_i$ and $u_{i+1} > c$. On the right side of Figure 1, the births happen at A, B, C, and E; on the left side, they happen at a and b.

We say that a *death* occurs at c if f_c has fewer components than $f_{c-\epsilon}$ for ϵ small. In our interpolated case, this happens at a point t_i where both $u_{i-1} < c = u_i$ and $u_{i+1} < c$. These happen at D, F, G, and H for the aggressive speed function, and at c and d for the normal one.

The main idea of persistence is to create a pairing of the births and deaths of f. To do this, we associate to each birth the height u_i where the component was born. Then, when we come to a death, we are merging two components. The *younger* of the two is the one with the higher birth value. We then say that the younger one dies at the merger since its component is merged with one that existed before it was born.

If a component was born at height u_i and dies at height u_j we record the pair (u_i, u_j) and say that the persistence of the component is $u_j - u_i$. Persistence measures how long a component existed. In the case we are discussing this can be interpreted as the size of the shape feature represented by the component.

For the aggressive speed function, the persistence pairs are $(A, H), (C, D), (E, F)$, and (B, G); for the normal one, they are (a, d) and (b, c).

B. Diagrams

All of the above birth, death, and persistence information is summarized compactly in the *persistence diagram* $D_0(f)$. This is a multi-set of dots in the plane, containing one dot (u_i, u_j) for each component born at u_i and killed at u_j. See Figure 2 for the persistence diagrams of the two speed functions from Figure 1.

We note that these diagrams allow one to hunt for some common functional motifs among a function population withot any need for pre-alignment in either the horizontal or vertical direction. For example, the question of whether two functions have a min-max pair of a certain size, independent of where the critical points occur in the interval domains, can be settled simply by looking at their respective persistence diagrams. Of course, this process can be automated, as described in the next section.

C. Stability

An important feature of persistence diagrams $D_0(f)$ is that they are robust to small changes in the input function f.

To make this precise, we need a metric on the set of persistence diagrams, so we now define the *Wassertein* distance $W_p(D_0(f), D_0(g))$ between two diagrams. First, we fix some $p \in [1, \infty]$. We then adopt the convention that every diagram contains a dot of infinite multiplicity, and zero persistence, at every point (u, u) along the major diagonal in the plane. For each bijection $\phi : D_0(f) \to D_0(g)$, we define its cost to be

$$C_p(\phi) = \left(\sum_{u \in D_0(f)} ||u - \phi(u)||_p \right)^{\frac{1}{p}};$$

[1]Small enough that $u_j \notin [c - \epsilon, c)$ for any j.

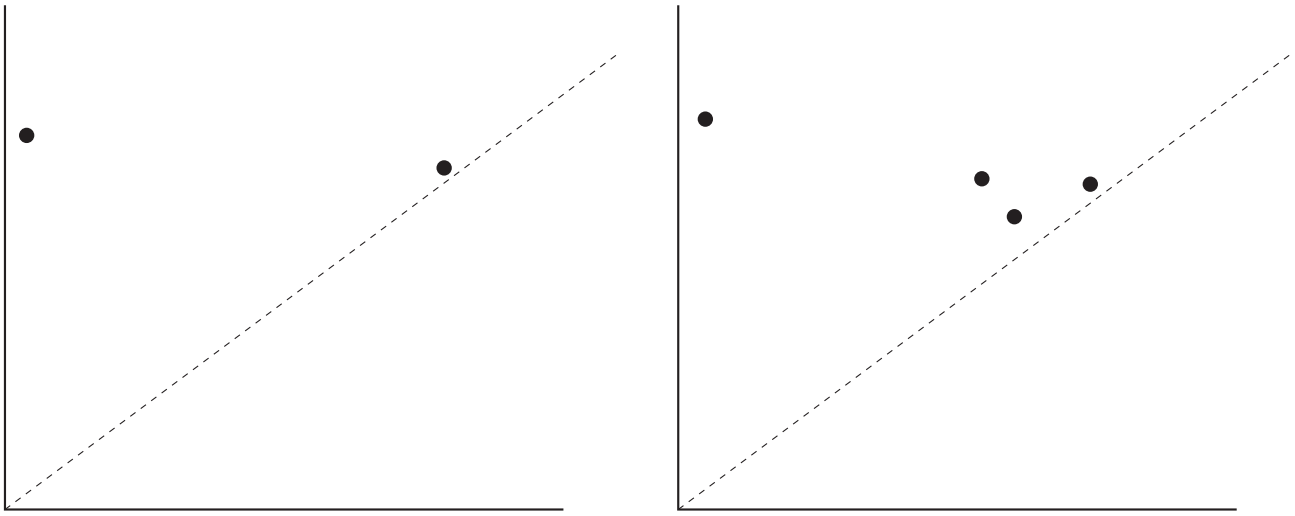


Fig. 2: The persistence diagrams $D_0(f)$ (left) and $D_0(g)$ (right), where f and g are the speed functions on the left and right, respectively, of Figure 1. Note that both diagrams have the same highest-persistence dot, which corresponds to their identical max speeds. The extra dots in the right diagram correspond to the extra variation in g.

note that such bijections exist even if the two diagrams have different numbers of off-diagonal dots, as we can always match extra dots to the diagonal dots in the other diagram. Finally, $W_p(D_0(f), D_0(g)$ is defined to be the minimum possible cost $C_p(\phi)$, as ϕ ranges over all possible bijections between the diagrams.

We remark that W_1 is often called the *earth-mover distance*, while W_∞ is the *bottleneck distance*. In addition, any of the distances W_p can be computed via a max-flow-min-cut algorithm.

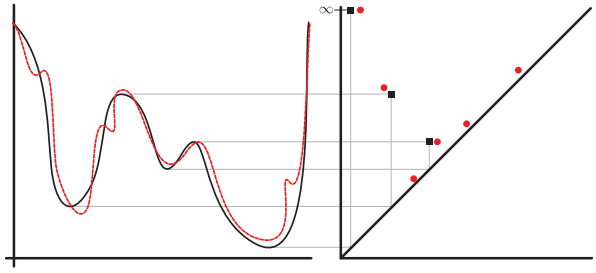


Fig. 3: Demonstration of stability. The graphs and persistence diagrams for a function and its noisy version are shown in black and red, respectively. The optimal bijection ϕ matches the high-persistence dots with each other, and the low-persistence red dots with the diagonal.

There are then several different stability theorems [8], [7], [9] which state that, under very mild conditions, $W_p(D_0(f), D_0(g)) \leq C||f-g||_\infty$. In other words, a small variation in the input function will not cause a large change in the output diagram. Figure 3 illustrates this phenomenon.

D. Augmented Morse filtration

In practice, we are given the finite set of values $\{f_i = f(t_i)\}_{i=0}^n$ and we compute the diagram using the discrete Morse filtration procedure. However, this algorithm discards a large amount of information concerning the values that the function attains. To avoid this waste, we augment the discrete Morse filtration procedure as described in Algorithm 1. This merely appends additional values to the diagonal of the the resulting persistence diagram. It should be noted that the stability results discussed in the last subsection also apply to these augmented diagrams.

The time complexity of both of these algorithms is dominated by the sorting procedure, and so these algorithms are $O(n \log n)$. The space complexity of these algorithms is $O(n)$.

IV. Learning Behavior from Diagrams

As we have seen, diagrams can encode the repetitive structure of behaviors. In order to exploit this encoding, we would like to project persistence diagrams into a Euclidean feature space so that distinct classes are rendered linearly separable. Ideally,

Algorithm 1 The (augmented) discrete Morse filtration algorithm.

Input $\{(i, f_i)\}_{i=0}^n$
Initialize diagram $D_0 \leftarrow \emptyset$, and class indicators $\{c_i\}_{i=0}^n \leftarrow \{-1\}_{i=0}^n$
Sort $\{(i_k, f_{i_k})\}_{k=0}^n$ so that $\{f_{i_k}\}_{k=1}^n$ is in ascending order
for all $k = 0, \ldots, n$ **do**
  $N \leftarrow \emptyset$
  if $i_k > 0$ **and** $c_{i_k-1} \neq -1$ **then**
    $N \leftarrow N \cup \{i_k - 1\}$
  end if
  if $i_k < n$ **and** $c_{i_k+1} \neq -1$ **then**
    $N \leftarrow N \cup \{i_k - 1\}$
  end if
  if $N = \emptyset$ **then**
    $c_{i_k} \leftarrow k$
  end if
  if $N = \{j\}$ **then**
    $c_{i_k} \leftarrow c_j$
    $D_0 \leftarrow D_0 \cup \{(f_{i_k}, f_{i_k})\}$ **IF AUGMENTED**
  end if
  if $N = \{i_k - 1, i_k + 1\}$ **then**
    $c_{\text{oldest}} \leftarrow \min(c_{i_k-1}, c_{i_k+1})$
    $c_{\text{youngest}} \leftarrow \max(c_{i_k-1}, c_{i_k+1})$
    for all $c_i = c_{\text{youngest}}$ **do**
      $c_i \leftarrow c_{\text{oldest}}$
    end for
    $D_0 \leftarrow D_0 \cup \{(f_{c_{\text{youngest}}}, f_{i_k})\}$
    $D_0 \leftarrow D_0 \cup \{(f_{i_k}, f_{i_k})\}$ **IF AUGMENTED**
  end if
end for
return D_0

the encoding of diagrams in this feature space should be sparse (i.e. the entries of the resulting feature vector should consist mostly of zeros). This reflects the desire that most objects should be sufficiently described using only a few of the atomic features represented by standard orthogonal directions in the feature space.

A. Binning and feature vectors

The work of Adcock et. al. ([1]) suggests a sophisticated approach towards constructing algebraic functions on the space of persistence diagrams which could then be used to produce features for machine learning. We pursue a far humbler approach, which we now describe, that turns out to be adequate for our particular needs.

For signals exhibiting repetitive structure, the visualization of persistence diagrams clearly indicates the spatial isolation of birth-death pairs. This suggests that we may employ regular tilings of the plane to "bin" the persistence diagrams, thereby obtaining sparsely-structured feature vectors. Moreover, when classes of signals exhibit distinct critical point behavior, projecting persistence diagrams in such a way induces linearly separable classes. We now describe the details of this projection. Given a persistence diagram $D = \{(\alpha_i, \beta_i)\}_{i=1}^N$, we first map to $\widetilde{D} = \{(\alpha_i, \beta_i - \alpha_i)\}_{i=1}^N$ as depicted in Figure 4. This skew-transformation just aligns the persistence diagram so that rectangular tilings of the upper-half plane are compatible with the natural constraints of the diagram.

Now, we fix the point sets $\{x_j\}_{j=1}^{r_h-1}$ and $\{y_i\}_{i=1}^{r_v-1}$ where

$$x_1 < x_2 < \cdots < x_{r_h-1}$$

and

$$0 < y_1 < y_2 < \cdots < y_{r_v}$$

The natural numbers r_v and r_h indicate the vertical and horizontal resolution of the "binned" diagram. The x values indicate the range of "birth" values that are under scrutiny, and the y value indicates the range of "lifetime" values that are under scrutiny. For simplicity, we define the auxiliary parameters

$$\Delta v_i = y_{i+1} - y_i \text{ and } \Delta h_j = x_{j+1} - x_j$$

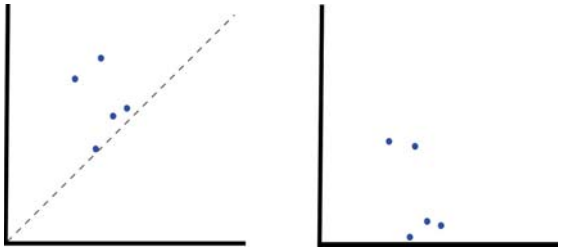


Fig. 4: Persistence diagram transformation.

for $i = 1, \ldots, (r_v - 2)$ and $j = 1, \ldots, (r_h - 2)$. to indicate the height and width of our rectangular bins. We also define the partition $\mathcal{I} = \{I_i\}_{i=1}^{r_v}$ of $[0, \infty)$ by

$$I_i = \begin{cases} [y_{r_v-1}, \infty) & i = 1 \\ [y_{r_v-i}, y_{r_v-i+1}) & 1 < i < r_v \\ [0, y_1) & i = r_h \end{cases}$$

and the partition $\mathcal{J} = \{J_j\}_{j=1}^{r_h}$ of $\mathbb{R}$ by

$$J_j = \begin{cases} (-\infty, x_1] & j = 1 \\ (x_{j-1}, x_j] & 1 < j < r_h \\ (x_{r_h}, \infty) & j = r_h \end{cases}$$

These partitions will induce a partition of the upper-half plane as depicted in Figure 5.

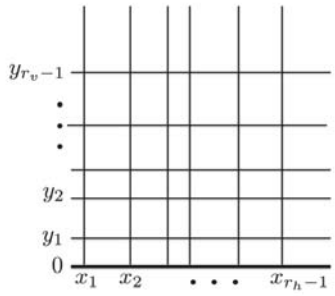


Fig. 5: Partitioning the upper-half plane for binning.

We now define the matrix $\text{bin}_\omega(\widetilde{D}) \in \mathbb{R}^{r_v \times r_h}$ (the space of r_v by r_h matrices) by setting

$$\left(\text{bin}_\omega(\widetilde{D})\right)_{i,j} = \left|(I_i \times J_j) \cap \widetilde{D}\right|$$

for all pairs (i, j) with $1 \leq i \leq r_v$ and $1 \leq j \leq r_h$. From this definition, it is clear that the partition elements I_1, J_1, and J_{r_h} are "overflow" regions that ensure the invariance of total counts under projection.

The savvy reader may note that this tiling of the plane for the purposes of constructing histograms is somewhat arbitrary. In this work, we have opted for the following strategy:

1) Fix a resolution r so that the binned diagrams are all r by r
2) For a given set of sampled, skewed diagrams, determine the partition nodes $\{x_i\}_{i=1}^{r-1}$ and $\{y_i\}_{i=1}^{r-1}$ so that x_i is the $100 \cdot (i/r)\%$ quantile of the empirical distribution over the birth values from this sample, and y_i is the $100 \cdot (i/r)\%$ quantile of the empirical distribution over the lifetime values from this sample.
3) Employ 10-fold cross-validation to choose r from a bounded range

There are of course many other ways in which one might perform this binning, but we have chosen this procedure for the sake of simplicity.

B. Models for binned diagrams

Here, we introduce several models for simulating and characterizing distributions over binned diagrams.

Logisitic regression: In this scenario, we are simply interested in determining how to separate the binned diagrams of distinct classes using hyperplane arrangements. For each class c, we have parameters θ_c and b_c. The probability of assigning an object with feature vector $\mathbf{x}$ to class c is then given by

$$p(c|\mathbf{x}) = \frac{\exp\{-\langle\theta_c, \mathbf{x}\rangle - b_c\}}{\sum_{c'} \exp\{-\langle\theta_{c'}, \mathbf{x}\rangle - b_{c'}\}}.$$

Training these parameters is done via the stochastic gradient descent procedure described in [23].

Poisson process: To more accurately reflect the fact that our feature vectors have integral entries, we may model the counts as arising from several independent Poisson-distributed random variables. Because the number of total counts of such a model can be variable, this model encodes a certain amount of domain invariance. The likelihood of a particular feature vector given a particular class under the Poisson model is given by

$$p(\mathbf{x}|c) = \prod p_{\text{Poisson}}(x_i|\lambda_{c,i}).$$

where $p_{\text{Poisson}}(x|\lambda) = \frac{\lambda^{-x}}{x!}e^{-\lambda}$. Classification is then performed using maximum likelihood estimation. Fitting the Poisson rates for each class given a set of labeled data is accomplished using the maximum likelihood estimate for λ_c. That is, given examples $X_1, \ldots, X_N$ from class c, we set

$$\lambda = \frac{1}{N}\sum_{i=1}^{N} X_i.$$

Multinomial: To capture the relative domain invariance of persistence diagrams more precisely, we may model the counts as a draw from a multinomial distribution where the number of classes is an input parameter. In this case, the likelihoods are given by

$$p(\mathbf{x}|c) = \binom{|\mathbf{x}|}{x_1, \ldots, x_K} \prod \theta_{c,i}^{x_i}$$

for parameter vectors $\theta_\mathbf{c}$ satisfying $\theta_{c,i} \geq 0$ for all i and $|\theta_c| = \sum \theta_{c,i} = 1$.

Fitting these parameters again proceeds using the maximum likelihood estimate, whence

$$\theta_c = \frac{1}{\sum_{i=1}^{N} |X_i|}\sum_{i=1}^{N} X_i$$

where $X_1, \ldots, X_N$ are the examples from class c.

C. Simulated Experiments

a) Description of simulated data: In this section, we describe the data used to train and test the models on our topological features. Some of the technical illustrations from this section can be found in the Appendix. For our experiments, we examine a data set of vehicular tracking information generated using SUMO, which has been primarily developed by the Institute of Transportation Systems at the German Aerospace Center. The road networks that the vehicles drove on were taken from the actual road network of Lubbock, but pruned down to a square of about 15 km on a side. The datasets consisted of tracks generated from simulation of 1000 vehicles split into two behavior classes: aggressive and normal. The distinct behaviors of these two classes of vehicles were obtained by feeding two sets of configuration parameters (impatience, top speed, etc.) into the SUMO simulation. We used all of SUMO's simulation capabilities including (but not limited to) traffic light simulator, passing, etc.

This data set consists of 1000 different vehicular paths, with each path consisting of around 360 triples of positional coordinates indexed by time. trained on sample target data generated using SUMO and exhibiting the two different driving behaviors obtained by modifying configuration parameters inside of SUMO that affect vehicle behaviors. Half of the vehicular paths exhibit "normal" behavior, while the other half exhibit "aggressive" behavior typified by speeding and accelerations. Typical speed profiles for the classes are illustrated in Figure 6. Clearly, the aggressive drivers tend towards higher speeds, though they seem somewhat less volatile. Persistence diagrams obtained form this procedure are illustrated in Figure 15. Binned diagrams for normal and aggressive classes are presented in Figures 16 and 17, respectively. It is clear that the binned diagrams provide very sparse features for the purposes of training classifiers.

To construct training and testing sets from this data, we first randomly split the tracks 800 training tracks and 200 test tracks. After fixing a window size N, we randomly subsample these tracks to obtain training and test tracklets of length N. Using this procedure, it is possible to get up to 10,000 representative samples. Once these sets have been constructed, we apply feature functionals to the individual tracklets to obtain data of the form (X, Y), where X is the binned diagram obtained from the feature functional and Y is the class label of the data (normal or aggressive). Data of this form is then used to train and test the accuracy of the model with respect to classification.

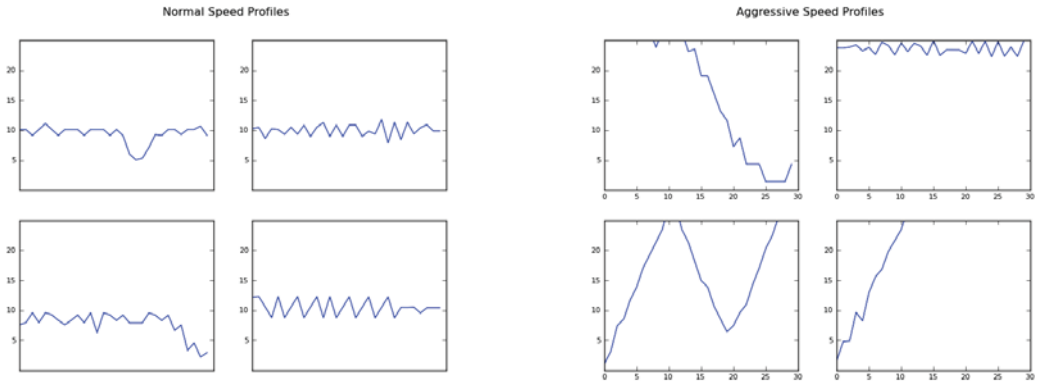


Fig. 6: Speed profiles of normal drivers (left) and aggressive drivers (right).

b) Experimental results: First, we examined the behavior of our classifiers for the speed (Figure 18), acceleration (Figure 19), and turning (Figure 20) feature functions described in Section II. In addition, we considered using all these features combined in Figure 21. From the training results, we see that the speed function provides the best test error, and the combined features only marginally improves the classification error. Once may clearly see that these test errors are stabilizing as the training size increases, which indicates that we are converging rapidly on the generalization errors of the models. Our very best result is a less than 2 percent test error.

To emphasize the invariance properties of the persistence diagrams with respect to homeomorphisms, we consider situations where the length of the input windows (that is, the number of speed values used to compute the different diagrams) varies over both training and testing. We consider window lengths of size 5, 10, 15, 20, 25, and 30 for both training and testing. The clear takeaway is that training can be done on relatively small sizes, but classification error always suffers in the low sampling regime. In order to validate the stability of our results, we have computed this array of errors for 25 randomized partitions of the data into training and test sets. Over these 25 random partitions, the standard deviation of these errors (which are random because the partition of the dataset is random) is uniformly bounded by 10^{-28}.The results are depicted in Figures 22, 23, and 24.

Finally, we illustrate the the general structural difference between the two classes by presenting the rates of the learned Poisson model (average binned diagram from each class) in a logarithmic scale in Figure 7.

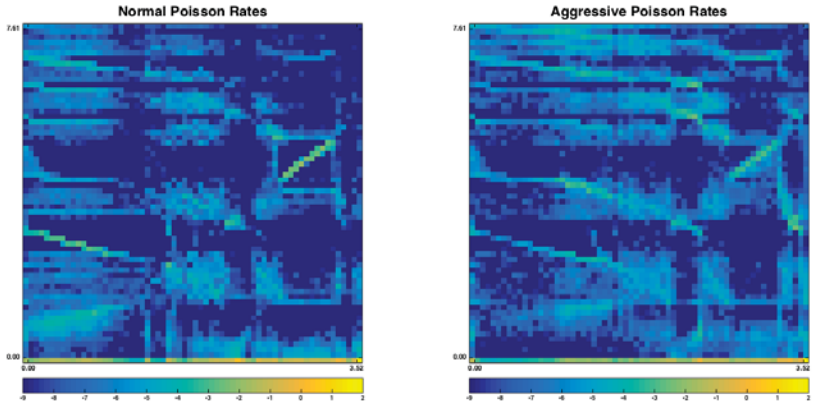


Fig. 7: Learned Poisson rates on a logarithmic scale illustrate structural distinctions between topological features exhibited by the two different classes. These images are obtained by taking the mean binned diagram over the two classes, and then applying a logarithm to each value in the resulting array to magnify distinctions. Lighter colors indicate higher concentrations of points in the persistence diagram. Strong lines illustrate common "motifs" exhibited by the persistence diagrams of the two classes. Note the subtle differences in the smearing patterns of these motifs between the two classes. indicate different structure in the volatility of the persistence diagrams. In general, the aggressive class is smeared more strongly upwards and leftwards, indicating that persistent components are born later and live longer. Comparing the bottom right corner of the images, we see that the short-lived persistent components live a little longer for the aggressive class, indicating that high-speed noise generally has a larger variance for the aggressive class. These kinds of subtle distinctions ultimately lead to the success of the classifiers.

V. The Tracker

A. The Baseline Tracker

The core framework of our approach is an MHT tracker which has been used for field-scale applications in areas including mobile missile detection systems, maritime ship tracking, and space situational awareness. This tracker (described in [19]) was developed for Upstream Data Fusion (UDF) to recover information in upstream data that would otherwise be lost using traditional downstream fusion methods. The sensor systems feeding the tracker are shown in the upper left part of Figure 8 with upstream data taps indicated.

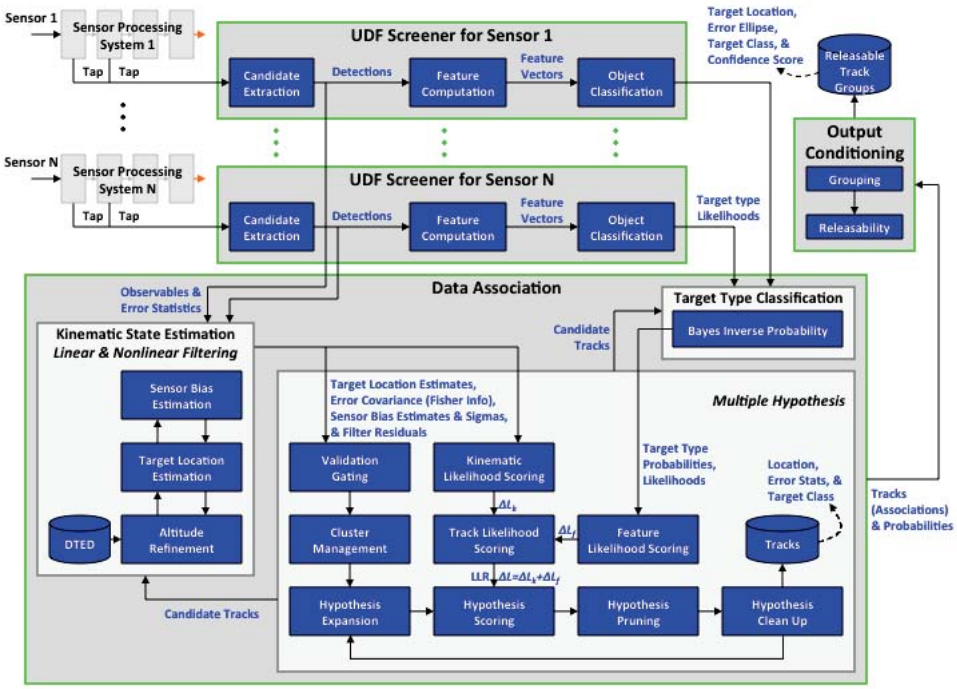


Fig. 8: Upstream Data Fusion (UDF) MHT tracker

The first step in processing is to screen the data to produce observables for kinematic state (e.g., position, velocity and acceleration states) estimation and target type features for target type estimation. The kinematic state estimation is accomplished with a Square Root Information Filter (SRIF) that uses Gauss Newton methods for iterative nonlinear filtering, compensates for sensor system biases, and for ground targets uses Digital Terrain Elevation Data (DTED). The SRIF and the target type estimation are obtained by analyzing the posterior in the Bayesian inference problem.

Data association for tracking can be viewed as a hierarchical Bayesian inference problem, where the kinematic estimation and type estimation yield parameters inside data association as shown in Figure 8. We use this approach to incorporate behavior estimation in MHT by adding another inverse problem for target behavior estimation to the nest in a later section.

The multiple hypothesis block of Figure 8 expands multi-track hypotheses, scores them with the LLR and prunes the hypotheses down to a reasonable number to carry forward. Obviously incorrect data associations are eliminated in a gating process if a track has a statistically excessive residual fit error. Also, data are clustered into disjoint regions of confusion in a divide and conquer approach.

The basis for the LLR can be found in Reid's fundamental expression in Equation (1):

$$\begin{aligned} p\left(\Omega_i^k|Z^k\right) &= p\left(\Psi_h, \Omega_g^{k-1}|Z(k), Z^{k-1}\right) \\ &= \frac{p\left(Z(k)|\Psi_h, \Omega_g^{k-1}, Z^{k-1}\right) p\left(\Psi_h|\Omega_g^{k-1}, Z^{k-1}\right) p\left(\Omega_g^{k-1}|Z^{k-1}\right)}{p\left(Z(k)|Z^{k-1}\right)}, \end{aligned} \tag{1}$$

where Ω_i^k is the data association hypothesis, Ψ_h is the current scan data association hypothesis, Ω_g^{k-1} is the hypotheses through $k-1$ scans, and $Z(k) = \{z_1(k), z_2(k), \ldots, z_{m_k}(k)\}$ is the set of measurements on the current scan at time t_k.

The solution recursively processes a scan of data to update the probabilities of data association hypotheses. A scan of data satisfies the scan constraint that, for each track in the hypothesis, there is at most one measurement in the scan and, for each measurement in the scan, there is at most one track in the hypothesis. Typical imagery data and Ground Moving Target

Indicator (GMTI) data satisfy the scan constraint. Typical signal data do not satisfy the scan constraint so the recursion is done for one measurement at a time. The left-hand side of the equation is the probability of the multiple track data association hypothesis given the data through time k. The first two terms in the numerator of the right-hand side of the equation are the conditional Bayesian likelihood terms. The left term of the two is the conditional likelihood function that is the probability density of the new data given the parent and child data association hypotheses and the previous data. The right term of the two is the Bayesian term that is the probability of the child given the parent and previous data. Since the current data does not enter this term, the probability of how the current data associates is based entirely upon prior information. Thus, it is a Bayesian probability computed from prior quantities such as the probability of detection and false alarm rate of the sensor system. It is easy to see that the highest probability hypothesis is the one with the highest product of conditional Bayesian likelihood terms over time. Taking the natural log of this and normalizing it produces the LLR scoring term for association hypotheses described by Bar Shalom, et al. in [2]. Assuming data in the scan are statistically independent of each other and that the kinematic space data and target type feature space data are independent of each other, the likelihood term can be expressed as

$$\begin{aligned} p\left(Z(k)|\Psi_h,\Omega_g^{k-1},Z^{k-1}\right) &= \prod_{j=1}^{m_k} p\left(z_j(k)|\Psi_h,\Omega_g^{k-1},Z^{k-1}\right) \\ p\left(z_j(k)|\Psi_h,\Omega_g^{k-1},Z^{k-1}\right) &= p\left(z_{ks,j}(k)|\Psi_h,\Omega_g^{k-1},Z^{k-1}\right) \\ &\quad \times p\left(z_{fs,j}(k)|\Psi_h,\Omega_g^{k-1},Z^{k-1}\right) \end{aligned} \tag{2}$$

The natural log of the kinematic space term can be computed using SRIF quantities as described in [5] to be the following

$$-\frac{1}{2}\left(e_k^t e_k + 2\frac{\log\det R_{k/k}}{\log\det R_{k/k-1}}\right) \tag{3}$$

The e_k term is the SRIF residual data fit error and the $R_{./.}$ term is the square root of the information matrix where the information matrix is the inverse of the covariance of the state estimate error.

For target type feature data, suppressing the association hypothesis, assume a finite number of exhaustive, mutually exclusive discrete target types and sum a joint distribution over the target types to obtain the likelihood as shown in the first expression in Equation (4). But the joint distribution can be expressed using Bayes rule in terms of the conditional likelihood conditioned

$$\begin{aligned} p\left(z_{fs,k}|\Psi_h,\Omega_g^{k-1},Z_{fs}^{k-1}\right) &= \begin{cases} p(z_{fs,k}|FT) & \text{for false target} \\ \sum_{i=1}^{N_T} p(z_{fs,k},T_i|Z_{fs}^{k-1}) & \text{for detected target} \end{cases} \\ \sum_{i=1}^{N_T} p(z_{fs,k},T_i|Z_{fs}^{k-1}) &= \sum_{i=1}^{N_T} p(z_{fs,k}|T_i,Z_{fs}^{k-1})p(T_i|Z_{fs}^{k-1}) \\ p(T_i|Z_{fs}^{k}) &= \frac{p(z_{fs,k}|T_i,Z_{fs}^{k-1})p(T_i|Z_{fs}^{k-1})}{p(z_{fs,k}|Z_{fs}^{k-1})} \end{aligned} \tag{4}$$

on the target type and the probability of the target type given past data as shown in the second expression in Equation (4). Further, the probability of the target type can be provided by the posterior shown in the last expression in Equation (4). The key quantity needed in the above is the conditional likelihood that can be determined using machine learning methods as described in [19].

The target type feature data can actually be target type classification computed using only the current data. Then the conditional likelihood above is the confusion matrix of the classifier as described in [3]. An upstream solution is to use target type features directly, rather than actually computing a target type classification from the current data in a track. This recursive solution provides a target type classification using all the feature data in the track.

It can be shown that the likelihoods provided by Equation (3) and Equation (4) are mathematically equivalent by deriving Equation (3) in terms of the integral over the continuous kinematic state space of the product of the conditional likelihood with the probability density of the kinematic state conditioned on the past data.

The experiments from Section IV indicate that our topological behavior features provide a net benefit when detecting agent behavior. In this section, we describe how we integrate these features and models into an improved MHT.

B. Augmenting the Tracker with Topoogical Features

Here, we depict our integration of the UDF MHT tracker with behavior classification using topological features. The block diagram in Figure 9 shows how the UDF MHT tracker of Figure 8 is generalized to include behavior classification.

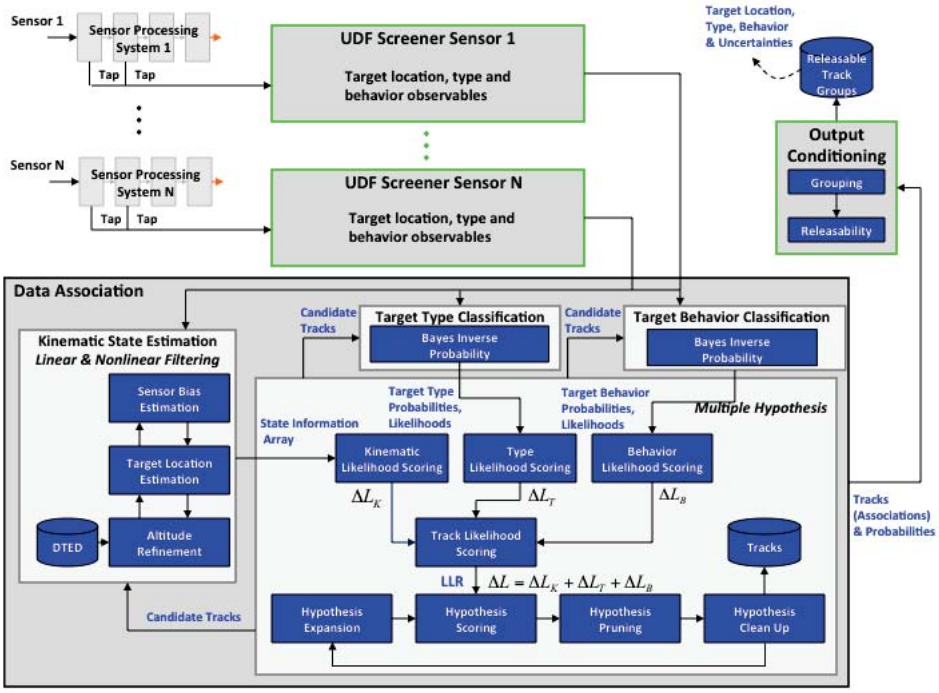


Fig. 9: UDF MHT tracker using topological features

The screener blocks and the multiple hypothesis block have been simplified to focus attention on the changes made to incorporate behavior classification. An additional inverse problem for target behavior classification has been included in the nested inverse problem formulation for data association. Changes in the LLR score for a track due to the current kinematic, type, and behavior data, respectively, are summed to obtain the total track score. This means that data are associated based on the consistency of target kinematic, type, and behavior data.

The equations that incorporate the target topological features are analogous to Equations (2) and (4) for target type features. The target type feature space data are replaced by target behavior topological feature space data, and target types are replaced with target behaviors. As mentioned previously, the recursive updating is a multiple rate process where the behavior LLR updates are done at a slower rate than the kinematic and type updates. The topological features are based on behavior functions such as speed, acceleration, and turning. The behavior topological features are computed over an interval of time sufficient for the behavior to be observed. Then an LLR update is computed.

It is assumed in Equation (2) that the feature data are statistically independent of the other data. This is also assumed for topological feature data. Although speed, acceleration, and turning are computed from kinematic data, it is not expected that the overall topological shape features will have much correlation with SRIF residuals. Also, the behavior functions considered here are computed from SRIF kinematic state estimates. Consequently, the estimate errors need to be taken into account when performing machine learning to determine conditional likelihood functions.

C. Simulated Experiments

c) Description of simulated data: A traffic scenario was constructed using SUMO to test the efficacy of our procedure in a reasonably realistic situation. In the scenario, two vehicles approach an intersection, stop for several seconds, and then pass through the intersection. Since target detection and measurement uncertainty typically degrade when vehicles stop moving with respect to the background environment, the period of the time when both vehicles stop at the intersection introduces an opportunity for a track swap by a tracker processing measurements collected by a sensor observing the scene, .

The two vehicles in the scene presented different driving behaviors. One vehicle exhibited a normal driving behavior, whereas the other vehicle exhibited the behavior typified by an aggressive driver. In this scenario, the normal driver obeys posted speed limits and demonstrates prudence when both accelerating and decelerating. On the other hand, the aggressive driver rapidly accelerates and decelerates and frequently exceeds the posted speed limit. These behaviors were realized by setting parameters defining the vehicle type in SUMO.

A simulated passive electro-optical (EO) sensor placed 9 km above the two vehicles provided angle-only measurements (i.e., horizontal and vertical angles relative to the sensor boresight) of the vehicles. The sensor nominally reports measurements of the targets at a 2 Hz rate. Simulations were run with different levels of measurement uncertainty to assess the sensitivity of the

association decisions to the sensor accuracy. We simulate an EO sensor that reports measurements that describe the line-of-sight to the detection using two angles (i.e., vertical and horizontal) relative to the sensor's boresight. Naturally, the measurement uncertainty is characterized by the angular uncertainty, and the uncertainty is the same for the measured vertical and horizontal angles. The measurement uncertainty is modeled as a independently and identically distributed zero-mean Gaussian random process with standard deviation σ_θ. The value of σ_θ is chosen to introduce position errors characterized in terms of the arc-length $r\sigma_\theta$, where r is the nominal distance of the sensor from the vehicles (i.e., 9 km for the scenario assessed).

d) Experimental results: Our simple scenario is illustrated. in Figure 10. Of the two depicted vehicles vehicles, one has a driver who is a speeder and one who is a normal driver. To isolate ideal performance of our procedure, we do not incorporate data from other vehicles present in the scene. This ideal performance should certainly degrade when the baseline tracker is loaded with information coming off of a complicated scene.

Sensor detection degrades as vehicles slow and stop, for instance, at an intersection. Tracking results will be shown for one example with and without topological features that focus on the difficulty of successfully tracking through an intersection. If a vehicle of interest has been identified but cannot be tracked through an intersection, then it will be lost. Furthermore, Monte Carlo results will be shown comparing performance in tracking through the intersection with and without topological features with varying levels of angular measurement errors.

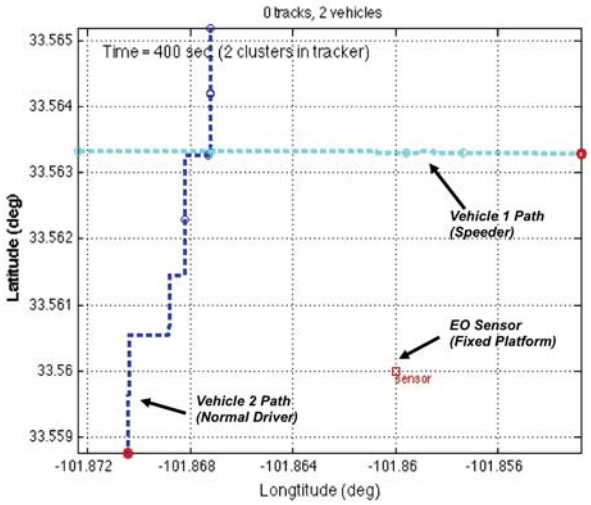


Fig. 10: Scenario with a speeder and a normal driver observed by airborne EO.

Tracking is shown in Figures 11 through 13 using MHT with and without topological features. The MHT is run with a maximum of 10 multi-track hypotheses to provide the ability to carry enough alternative hypotheses to correct past association mistakes as new data clarifies association. The logistic regression approach described previously was used to perform machine learning to produce a behavior classifier. The target behavior classification provided by the topological features is shown as the probability of speeder where a low probability indicates a normal driver and a high probability indicates a speeder. To emphasize that the tracker without topological features cannot classify driver behavior the probability of speeder without topological features is set to 0.5.

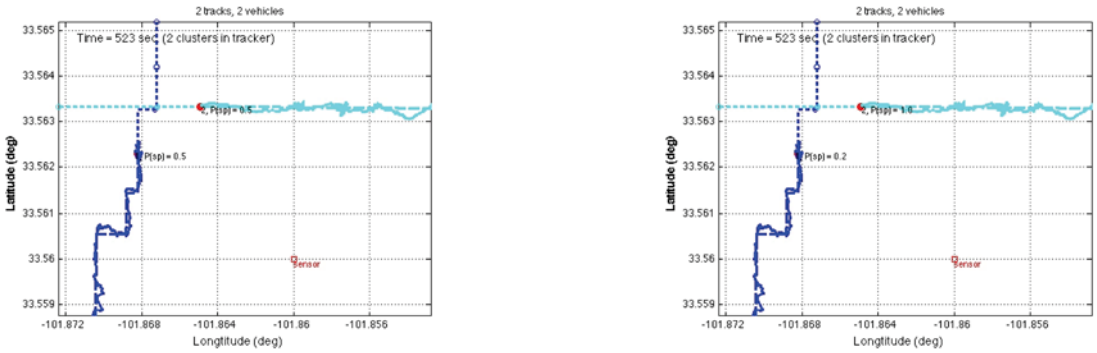


Fig. 11: Trackers before intersection.

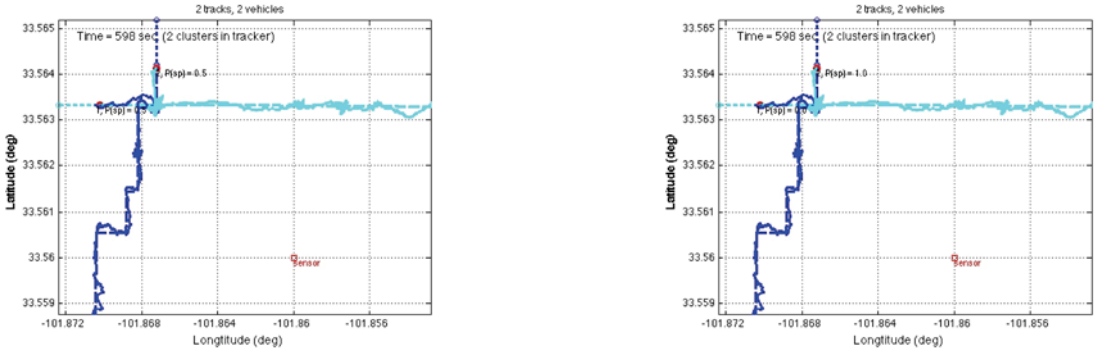


Fig. 12: Trackers after intersection but before LLR update.

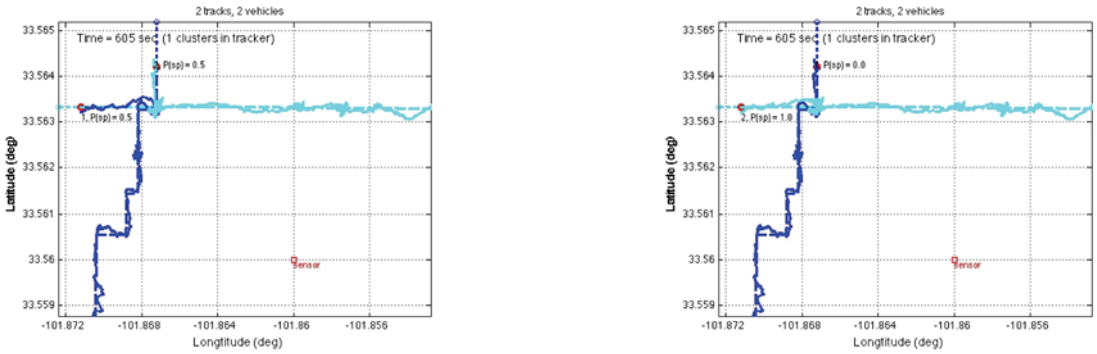


Fig. 13: Trackers after intersection and LLR update.

In Figure 11, the tracker results are shown prior to the intersection. The tracks are the same with and without the topological features because there is little confusion for these tracks before the intersection. But the tracker with topological features has accurate estimates of driving behavior. In Figure 12, the tracking picture is shown a short time after the intersection before enough time has passed for a topological feature update to the LLR score. Both trackers have the same incorrect tracks that switch vehicles at the intersection. It is in Figure 13 that the difference occurs. MHT with topological features is able to correct the association error when it updates the LLR after the intersection. It estimated driver behavior before the intersection and as the behavior emerged after the intersection it could use the information from the behavior to correctly associate the data. Without topological features the tracker could not correct itself.

In order to establish the statistical performance with and without topological features, Monte Carlo results are shown in Figure 14 for varying levels of measurement uncertainty. For each measurement uncertainty, 100 trials were performed and statistical error bars are shown. It is clear that association is better with topological features. In fact, topological features allow tracking to be successful at substantially higher levels of measurement noise.

VI. Conclusion

We have shown that persistence, combined with learning, provides a useful tool for measuring agent activity, and we have demonstrated how to apply that measure to improve image-based tracking. There are a large number of ways that this methodology can be generalized, including the use of more sophisticated measurement functions of driving behavior. For example, we tested the use of delay reconstruction to transform functions into point clouds - the results can then be analyzed with a variety of data analysis tools. Whatever measure is used, one can train the weighting of topological features on different environments, including different cities, rural or mountainous environments, etc. And one can include other types of intelligence such as cell-phone signatures, tweets, etc. to improve tracking performance.

Another important generalization of the ideas in this paper is the capture of *collaborative behavior* among a collection of agents. This behavior is more likely to exhibit itself at the large scale, as drivers act in parallel and make interesting driving patterns (departing from the same point and remerging later, etc.). In many ways, this is what persistent topology should be best at, given that it looks at large scale behavior rather than the fine details, yet still provides quantitative measurement of behavioral patterns. To do this will require the use of higher dimensional persistence, so this is beyond the scope of the current paper. We have already explored the use of one-dimensional persistence to track “inefficient driving” or “loopiness” [17], stopping times, and other behaviors that indicate nonchalance. Many other such patterns are detectable in this way.

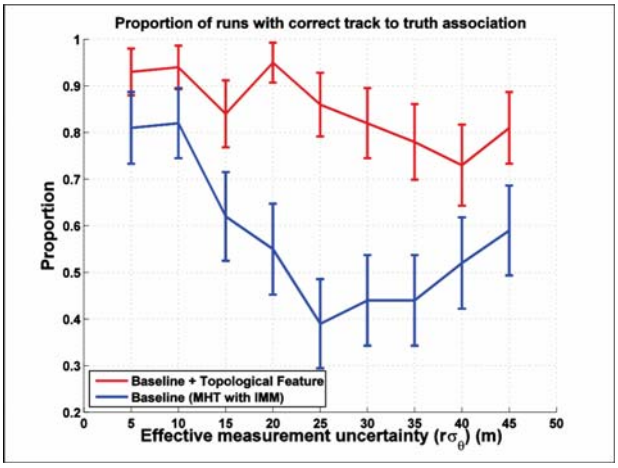


Fig. 14: Monte Carlo association performance as measurement uncertainty varies.

REFERENCES

[1] A. Adcock, E. Carlsson, and G. Carlsson. The ring of algebraic functions on persistence bar codes. *ArXiv e-prints*, 2013.

[2] Y. Bar-Shalom, S.S. Blackman, and R.J. Fitzgerald. Dimensionless score function for multiple hypothesis tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(1):392–400, January 2007.

[3] Y. Bar-Shalom, T. Kirubarajan, and C. Gokberk. Tracking with classification-aided multiframe data association. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(3):868–878, July 2005.

[4] Paul Bendich, David Dunson, Bailey Fosdick, John Harer, and Nate Strawn. Statistical inference with topological features. 2014. manuscript.

[5] G.J. Bierman, M.R. Belzer, J.S. Vandergraft, and D.W. Porter. Maximum likelihood estimation using square root information filters. *Automatic Control, IEEE Transactions on*, 35(12):1293–1298, Dec 1990.

[6] S.S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[7] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 237–246, New York, NY, USA, 2009. ACM.

[8] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, January 2007.

[9] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l_p-stable persistence. *Found. Comput. Math.*, 10(2):127–139, February 2010.

[10] Mary-Lee Dequéant, Sebastian Ahnert, Herbert Edelsbrunner, Thomas M. A. Fink, Earl F. Glynn, Gaye Hattem, Andrzej Kudlicki, Yuriy Mileyko, Jason Morton, Arcady R. Mushegian, Lior Pachter, Maga Rowicka, Anne Shiu, Bernd Sturmfels, and Olivier Pourquié. Comparison of pattern detection methods in microarray time series of the segmentation clock. *PLoS ONE*, 3(8):e2856, 2008.

[11] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454 –463, 2000.

[12] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[13] P. Frozini and B. Landi. Size theory as a topological tool for computer vision. *Pattern Recognition and Image Analysis*, **9**:596–603, 1999.

[14] Jennifer Gamble and Giseon Heo. Exploring uses of persistent homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, 101(9):2184 – 2199, 2010.

[15] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[16] James Llinas, David Hall, and Martin Liggins. *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC Press, 2009.

[17] Elizabeth Munch. *Applications of Persistence Homology to Time Varying Systems*. PhD thesis, Duke University, 2013.

[18] James R. Munkres. *Elements of Algebraic Topology*. Addison Wesley, 1993.

[19] A.J. Newman and G.E. Mitzel. Upstream data fusion: History, technical overview, and applications to critical challenges. *APL Technical Digest*, 31(**3**), 2013.

[20] José A. Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *To appear*, 2013.

[21] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 24(**6**):843–854, 1979.

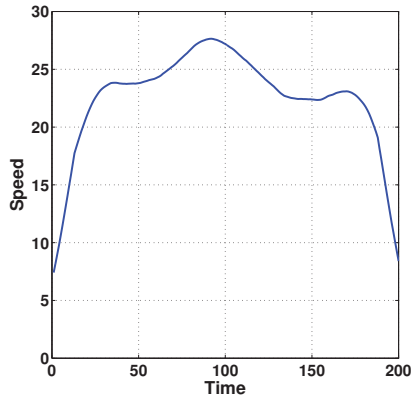
[22] Torbjorn Wigren. Target-type probability combining algorithms for multisensor tracking. volume 4380, pages 46–62, 2001.

[23] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

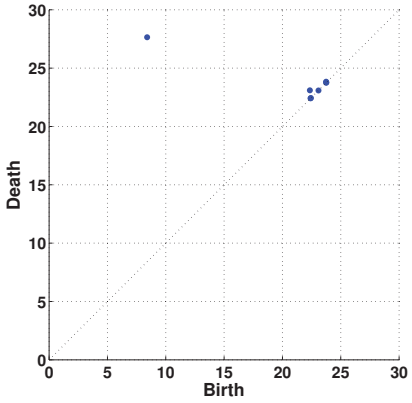
ACKNOWLEDGMENT

This work was partially supported by the grants OASIS JHU Subcontract 110279 on HQ0034-12-C-0024, DARPA LIMS Subcontract on HDTRA1-11-1-0048, and NSF-DMS grant 10-45153.

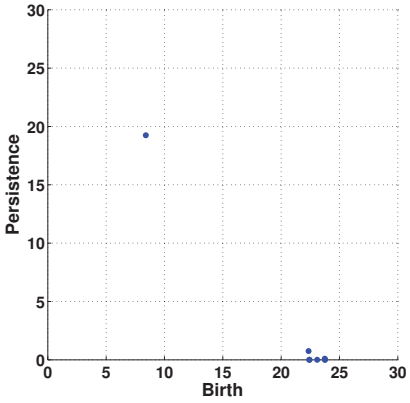
APPENDIX
ADDITIONAL FIGURES



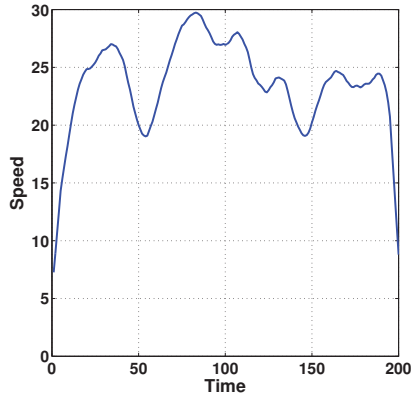
(a) [Example speed profile for a normal driver]



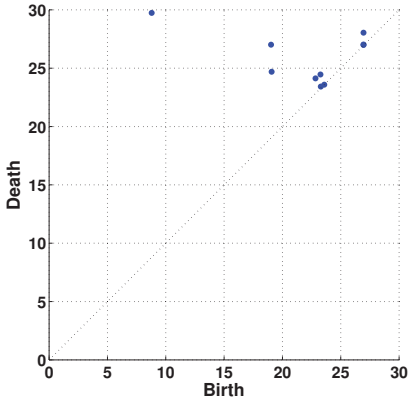
(b) [Persistence diagram derived from the profile in Figure 15a]



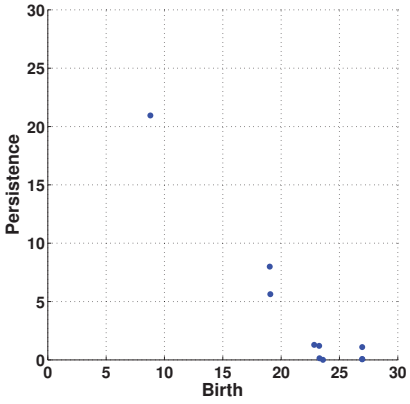
(c) [Transformation of persistence diagram in Figure 15b]



(d) [Example speed profile for an aggressive driver]



(e) [Persistence diagram derived from the profile in Figure 15d]



(f) [Transformation of persistence diagram in Figure 15e]

Fig. 15: Topological data analysis of speed profiles corresponding to normal (top row) and aggressive (bottom row) driving behaviors. Normal drivers obey posted speed limits, demonstrate prudence when both accelerating and decelerating, and avoid interfering with other vehicles. Aggressive drivers rapidly accelerate and decelerate, exceed the speed the limit, and frequently impede other vehicles. Persistence diagrams (Figures 15b and 15e) are derived from the speed profiles corresponding to normal and aggressive driving behaviors (Figures 15a and 15d). A *birth* value identifies the local minimum of the function that initiated a new component, and a *death* value identifies the local maximum of the function that merged two components. A component is a closed interval in the domain of the function. The death value is paired with the larger of two local minima representing the two components merged. The transformed persistence diagrams (Figures 15c and 15f) show the *persistence* corresponding to the component birth value, which is the difference between the birth and death values.

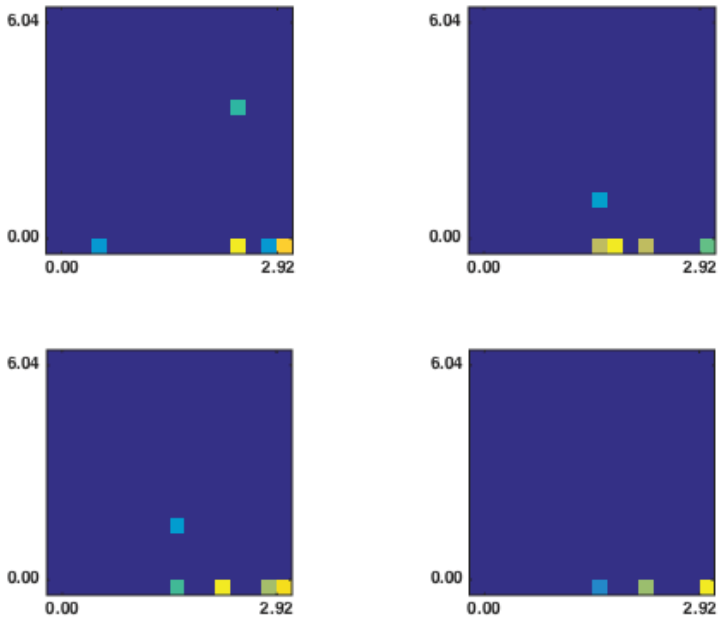


Fig. 16: These are examples of binned diagrams used as training and testing features for the normal speed profiles, with 16 by 16 resolution. One immediately sees the sparsity of these features, as the dark blue indicates zero values. The lighter colored pixels indicate the relative size of bin counts, or the number of points in the birth-lifetime diagrams that land in the bin associated with the pixel index.

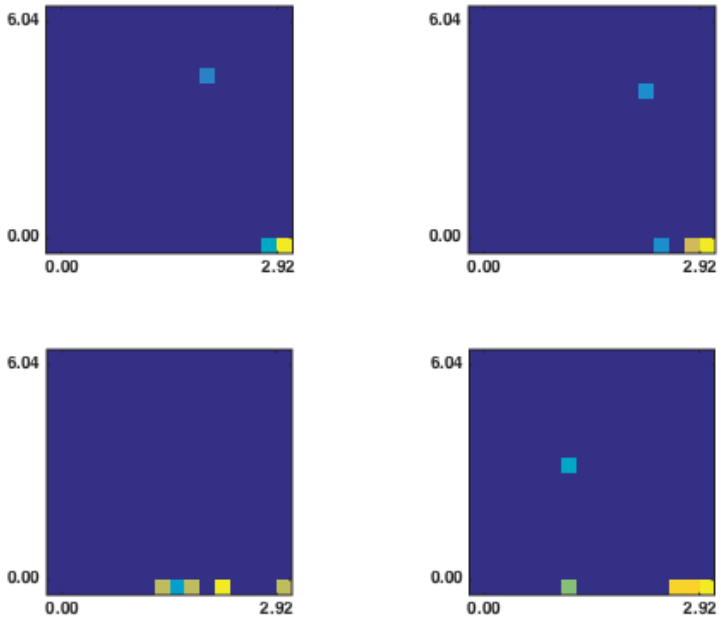


Fig. 17: This is the counterpart figure to Figure 16 illustrating binned diagrams for the aggressive speed profiles. We can see that there are some slight differences in the structure of the counts, and these diagrams generally have a point of higher persistence or several points with higher velocity.

Fig. 18: Test errors using binned diagrams computed from the speed function. The test errors of the logistic regression (red triangles), the Poisson model (blue squares), and the multinomial model (green circles) as a function of the size of the training data.

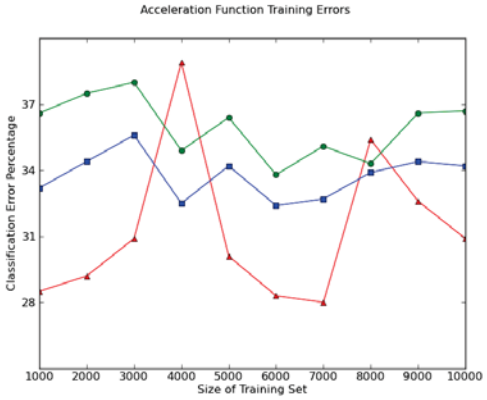


Fig. 19: Test errors using binned diagrams computed from the acceleration function. The test errors of logistic regression (red triangles), the Poisson model (blue squares), and the multinomial model (green circles) as a function of the size of the training data.

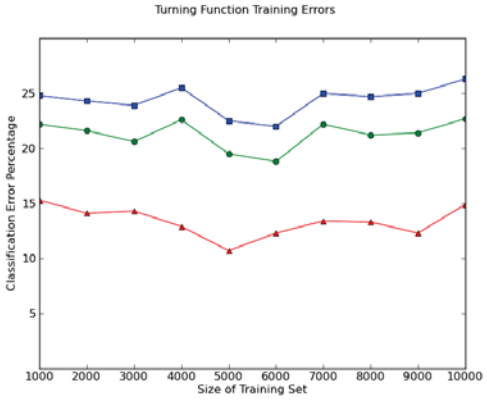


Fig. 20: Test errors using binned diagrams computed from the turning function. The test errors of logistic regression (red triangles), the Poisson model (blue squares), and the multinomial model (green circles) as a function of the size of the training data.

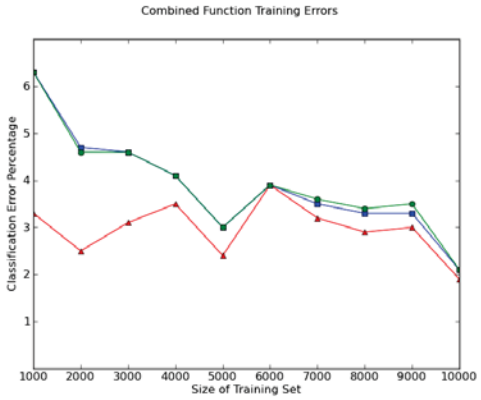


Fig. 21: Test errors using binned diagrams computed from the the speed, acceleration, and turning functions all combined. The test errors of logistic regression (red triangles), the Poisson model (blue squares), and the multinomial model (green circles) as a function of the size of the training data.

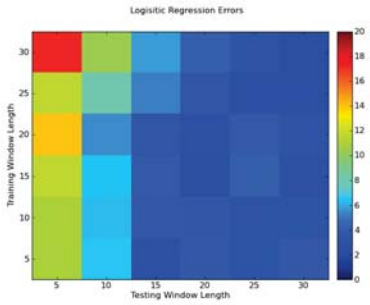


Fig. 22: Average classification percentage errors for logistic regression over 25 randomized training runs.

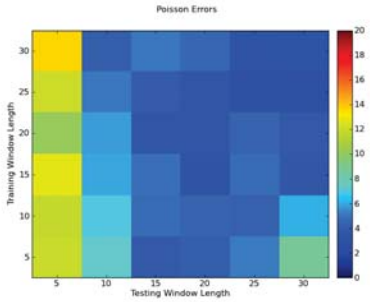


Fig. 23: Average classification percentage errors for classification under a Poisson model, with randomized input order over 25 runs.

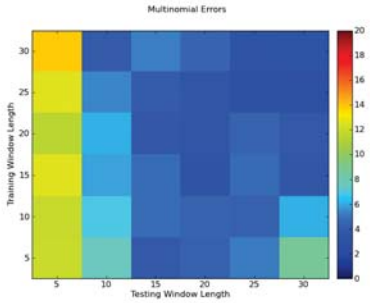


Fig. 24: Average classification percentage errors for classification under a multinomial model, with randomized input order over 25 runs.