Behavior Anomaly Detection

Kristofor B. Gibson

ABSTRACT

Modern warfare demands situational awareness of entities in the environment. To enhance the warfighter's situational awareness, we developed an algorithm that detects anomalous behavior in the warfare environment. Changes in entities' behavior can be an indicator that existing prediction models or assumptions must be updated to remain useful for decision-making. Specifically, we introduce a new classification method—sequential sample consensus (SeqSAC)—that identifies anomalous behavior based on a series of observations of an entity. SeqSAC can support a wide variety of models from simple to complex. We first demonstrate the utility of SeqSAC with a simple limited-degree-of-freedom kinematic model of a moving body, and then we demonstrate the ability to incorporate more complex models using the finite-state machine in Advanced Framework for Simulation, Integration and Modeling (AFSIM). Finally, we discuss the ability to extend SeqSAC to identify anomalies in coordinated entity behaviors.

INTRODUCTION

Chandola, Banerjee, and Kumar succinctly define anomaly detection as "finding patterns in data that do not conform to expected behavior."[1] A number of terms are used interchangeably to refer to deviations from expected behavior—"of these, anomalies and outliers are two terms used most commonly in the context of anomaly detection; sometimes interchangeably."[1] In detecting an anomaly, one must be able to observe a pattern and assess that pattern relative to a reference case of "normal" or "expected" behavior. Motor vehicle traffic offers a common example: an observed pattern—vehicles traveling in one direction—compares well with a mental model of "normal" traffic. However, a vehicle traveling in the opposite direction of the vehicles near it might, therefore, constitute anomalous behavior and mandate increased attention or decisions or actions.

A common approach to anomaly detection (see the extensive survey by Chandola, Banerjee, and Kumar[1]) requires data that contain both normal and abnormal (anomalous) behavior and that the normal behavior can be defined by a model with a set of parameters in one or more dimensions. Anomalies in a two-dimensional space are depicted in Figure 1. Two clusters of data that are considered "normal" or "inliers" are the regions

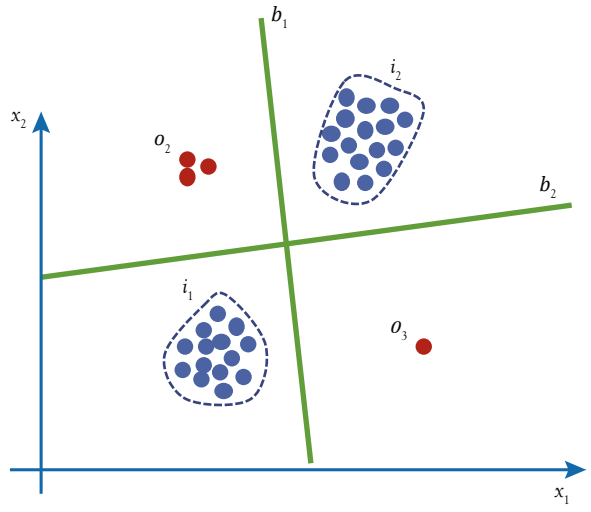

Figure 1. Example of two-dimensional data samples that are inliers and outliers (anomalies). The expected normal data points are labeled with i_1 and i_2. The anomalies are in the regions labeled o_2 and o_3. The boundaries, b_1 and b_2, represent possible decision boundaries.

labeled i_1 and i_2. The "outliers" or anomalies are reflected as data points far away from the inlier regions, and are labeled o_2 and o_3. The dimensions, x_1 and x_2, represent the observation space.

Detecting anomalies requires a model or knowledge of the expected normal behavior and a method of measuring distance from the expected normal regions. Figure 1 depicts a clear and simple case in which two decision boundary lines, b_1 and b_2, are positioned such that they separate the clusters into four distinct regions. For this example, normal behavior would be classified if the new sample falls within the upper-right or lower-left regions subtended by lines b_1 and b_2. Samples that fall within the upper-left or lower-right regions would be classified as anomalous. To revisit the example of vehicle driving behavior, consider the road direction parameter to be x_1 (e.g., $x_1 \gg 0$ = north, $x_1 \sim 0$ = south), and likewise the vehicle travel direction parameter to be x_2 (e.g., $x_2 \gg 0$ = north, $x_2 \sim 0$ = south). Similar to the example illustrated in Figure 1, a vehicle would be classified anomalous if traveling north on a southbound road.

The type of data set shown in Figure 1 is a uniquely tractable data set for anomaly detection because the expected normal regions are clearly defined and anomalies are also clearly separated from the normal regions with decision boundaries. However, the real world, and particularly modern warfare, often present scenarios where uncertainty and data scarcity demand a more robust approach to anomaly detection. When anomalous data points are exceedingly infrequent and boundaries are unknown or ambiguous, it becomes more difficult to develop and test anomaly detection algorithms. Collecting useful data in adequate quantities may be expensive or impossible, and assumptions may be required to characterize the regions of normal behavior.

To be adequately robust and useful, anomaly detection algorithms must be insensitive to changing environments. In the example of motor vehicle traffic, a useful anomaly detector should identify aberrant motorist behavior on all roadways (i.e., it should not be limited to cardinal directions). Additionally, a useful anomaly detector should be dynamic and support adjusting behavioral boundaries over time and space. By comparison, Figure 1 represents a mean ergodic case (i.e., clusters are independent of time) requiring only a static framework for anomaly detection.

In a combat scenario, and particularly in cases immediately preceding the commencement of hostilities, detecting anomalous behavior that might indicate malicious intent is particularly challenging because entities will likely exhibit "normal" behavior for as long as possible to avoid undue attention and thus preserve the element of surprise. Or an entity could behave in a way that deviates from "normal" or expected patterns specifically to confuse the combat decision-maker. Such "pollution" of the observed track history can add noise to the data set that can make it difficult to establish clear boundaries between normal and abnormal regions of the entity state-space.

In this article, we present a behavior anomaly detection algorithm for entities traveling in three-dimensional space over time (e.g., a track history of an aircraft). We relax the data requirements in the track history by assuming that the history provides only first-order dynamics (position and time). Thus, our problem is reduced to detecting a behavior change given only a history of locations and a behavior model (or models). It is assumed that the anomaly behavior is scarce in data or not known. We generalize our behavior anomaly detection algorithm such that it is robust to changing environments (i.e., can detect anomalies independent of time and location).

Building on anomaly detection methods that are application agnostic and rely heavily on an abundance of data, in recent work to detect anomalies from flight data, Janakiraman and Nielsen[2] explore using extreme learning machines using higher-level parameters. Our approach is an alternative to a more robust solution, one in which empirical data (e.g., track position) are exclusively used, and measured behavior parameters are hidden. Given multivariate, time-series data and a behavior model or models, we can identify anomalies using a unique sequential sample and consensus (SeqSAC) algorithm. We classify tracks as inlier or outlier with a small amount of data online and determine anomalies based on the first occurrence of outliers. This approach serves as a foundation for dynamic, simultaneous construction of complex behavior models that can then be applied to classify subsequent track behavior.

PROPOSED METHOD

In developing a behavior anomaly detector, our goal is to identify the location and time in data where and when an entity exhibited a behavior change that would be considered anomalous or would satisfy the criteria for alerting a human observer or autonomous systems to a potentially critical change. We accomplish this by using a behavior estimator that updates as more data points are observed (as time increases) and labels data as outliers when they do not support the estimated behavior. This sequential behavior is the foundation of our proposed method—the SeqSAC algorithm. This method is similar to the random sample and consensus (RANSAC) paradigm presented by Fischler and Bolles[3] wherein they iteratively sampled and classified inlier data to obtain a robust location determination. Similarly, given aircraft position and time data points, SeqSAC is used to determine whether the new data points are inliers or outliers. When inlier data points are classified, a more refined behavior model can be developed, against which future observations can be compared and anomalous activity detected. SeqSAC uses the inlier and outlier labeled data points to learn the behavior of the entity as new data arrive.

Introduction of Terms

First we present a few terms and functions to support the development of SeqSAC and its usefulness in behavior anomaly detection. Let the $3 \times M$ matrix $\boldsymbol{X}$ be the track history of an entity, where $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$. The kth column of $\boldsymbol{X}$ is the entity's three-dimensional position in time, $\boldsymbol{x}_k = (x_k^1, x_k^2, x_k^3)^T$[1]. (In this article, without loss of generality, we use the East-North-Up, or ENU, coordinate system for three-space positions.) The absolute time associated with the kth observation, $\boldsymbol{x}_k$, is $t_k \in \mathbb{R}$. In this article, we will model a behavior of an entity with a function f that maps a time t and $p \times 1$ parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{\boldsymbol{P}}$ to a position,

$$f_{\boldsymbol{\theta}} : t \longmapsto \boldsymbol{x} \in \mathbb{R}^3. \tag{1}$$

For convenience, we also allow the model to operate on a vector of time samples ($\boldsymbol{t}$) to generate a track history matrix, $\boldsymbol{X}$,

$$f_{\boldsymbol{\theta}} : \boldsymbol{t} \longmapsto \boldsymbol{X} \in \mathbb{R}^{3\times M}, \tag{2}$$

where $\boldsymbol{t} \in \mathbb{R}^M$. Note that it is assumed that if the track history in the matrix $\boldsymbol{X}$ is known, then the time series in $\boldsymbol{t}$ is known. The $\boldsymbol{X}$ is not known if only $\boldsymbol{t}$ is provided. Given the nth behavior model function, $f_{\theta_n}(\cdot)$, a companion behavior estimator, $g_n(\cdot)$, approximates the inverse function of $f_{\theta_n}(\cdot)$ and is needed to generate a best-guess parameter given a track history. The behavior model estimator in a general form is an approximation of the inverse model function,

$$f_{\theta_n}^{-1} \approx g_n : \boldsymbol{X} \mapsto \widehat{f_{\theta_n}}. \tag{3}$$

For this work, we use a weighted norm for minimizing a distance cost to represent the behavior model estimator,

$$g_n(\boldsymbol{X}) = \min_{f_{\theta_n}} \left\| w_k\left(f_{\theta_n}(t_k) - \boldsymbol{x_k}\right)\right\|_2^2 \forall\, k, \tag{4}$$

where in Eq. 4 the Euclidean distance between the kth model prediction and kth sample is minimized for all samples. The SeqSAC algorithm changes the values of w_k to find the inliers and outliers in the data, $\boldsymbol{X}$. We represent the weight values for each sample with the vector $\boldsymbol{w} \in \mathbb{R}^{M\times 1}$ and can more compactly denote Eq. 4 with

$$g_n(\boldsymbol{X}) = \min_{f_{\theta_n}} \left\| f_{\theta_n}(\boldsymbol{t}) - \boldsymbol{X}\right\|_{\boldsymbol{w}}^2. \tag{5}$$

SeqSAC Algorithm

The SeqSAC algorithm uses the same paradigm as the RANSAC algorithm described by Fischler and Bolles[3] where a consensus metric is used to determine which model best fits the data by sampling small subsets of the data. SeqSAC also uses an ϵ-ball metric, which defines the maximum distance a sample must be to be assigned as an inliear. But SeqSAC differs from RANSAC in two ways. First, the samples are not randomly selected at each iteration but are sequentially sampled over time with a sampling overlap. Second, as we show later, we can use SeqSAC recursively to build a sequence of behaviors that best match the track history.

RANSAC is useful in solving the problem of estimating a model when the choice of any data point has equal probability of contributing to the estimate. For example, a linear regression model can work with any two points selected at random. Estimating a behavior of an entity that has a time-dependent set of data is dependent on the relative time differences of selected samples. The behavior of the entity may change over time, introducing sample selection dependency when estimating a behavior. Therefore, RANSAC would randomly settle on an inlier set and outlier set; an entity that changes its behavior halfway through the data would cause RANSAC to choose the first half of the data to be either inliers or outliers 50% of the time if the random selection was uniformly distributed. SeqSAC—with sequential sampling over time—would be consistent in its inlier/outlier classifications, thus offering a consistent anomaly detection mechanism. Also, SeqSAC requires significantly fewer samples for search than RANSAC and will achieve the same results. If there are $N \gg 4$ total samples, with four samples minimum to estimate a model and for SeqSAC, a two-sample overlap, RANSAC's entire sample space would be $\binom{N}{4} = \frac{(N-3)(N-2)(N-1)N}{4!}$ samples. SeqSAC's sample space would only be $\frac{N}{2}$. So for this example,

RANSAC sample complexity is $O(N^4)$ and SeqSAC is $O(N)$.

In a basic form (no recursion), the inputs to SeqSAC are the track history, $\boldsymbol{X}$; a behavior model function, $f_{\boldsymbol{\theta_n}}$; and the behavior estimator function, g_n. The output from SeqSAC is a vector of weights $\boldsymbol{w}$, where each kth component, w_k, is either a value of 1 to indicate the kth data point, $\boldsymbol{x}_k$, is an inlier or a value of 0 to indicate the data point is an outlier according to the estimated model f_{θ_n}. All anomalies in $\boldsymbol{X}$ are all data points where $w_k = 0$. The first occurrence of the anomaly in time, t_a, is the earliest occurrence of $w_k = 0$:

$$t_a = \inf\{\boldsymbol{t} \mid w_k = 0\}, \tag{6}$$

where the $\inf\{\boldsymbol{t} \mid w_k = 0\}$ operator selects the minimum value within the vector, t, for only components associated with $w_k = 0$.

The recursive form of SeqSAC sequentially extracts samples from $\mathbf{X}$ with the previous weight vector $\mathbf{w}_{s-1}$ and proceeds to determine a consensus with only data points associated with $\overline{\mathbf{w}}_{s-1}$,

$$\boldsymbol{w}_s = \text{SeqSAC}(\mathbf{X}(\overline{\mathbf{w}}_{s-1}), f_{\theta_n}, g_n), \tag{7}$$

where the input data matrix, $\mathbf{X}(\overline{\mathbf{w}}_{s-1})$, indicates SeqSAC will only consider samples where $w_{k,s-1} = 0$. The recursion ends when the length of $\boldsymbol{w}_s$ is 0 or too small for g_n to make a valid estimate of f_{θ_n}. Since the inliers are known from the vector $\boldsymbol{w}_s$, a robust estimate of $f_{\boldsymbol{\theta_n}}$ comes free with the SeqSAC algorithm by using $\hat{f}_{\theta_n} = g_n(X(\boldsymbol{w}_s))$.

SeqSAC algorithm details are presented at the end of this article. The following sections provide applications of SeqSAC with multiple behavior types.

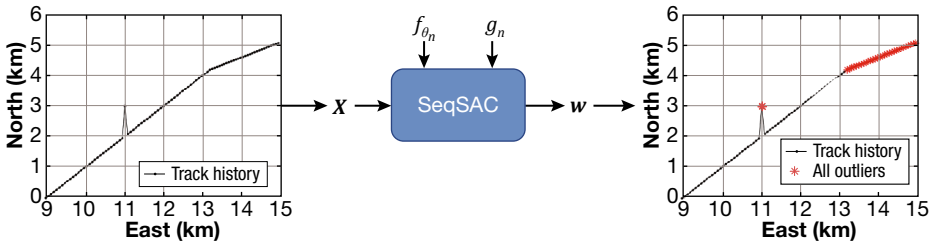

Figure 2. SeqSAC fundamental algorithm overview. The left plot is the input track history of a single entity that is traveling northeast. The input parameters to SeqSAC are $\boldsymbol{X}$, f_{θ_n}, g_n and the output is the vector $\boldsymbol{w}$ that classifies all data in $\boldsymbol{X}$ as an inlier or outlier.

SeqSAC with Polynomial Trajectory Example

In this section, we use a polynomial trajectory model to demonstrate SeqSAC. The trajectory model,

$$f_{\boldsymbol{\theta_p}}(\boldsymbol{t}) = \boldsymbol{\theta_p^T}[\mathbf{1}, \boldsymbol{t}, \boldsymbol{t^2}] = \boldsymbol{\theta_p^T S}, \tag{8}$$

is dependent on $\boldsymbol{\theta_p} \in \mathbb{R}^{\mathbf{3\times1}}$, which is the parameter vector that defines the initial position, velocity, and acceleration. (The column vector, $\boldsymbol{t^2}$, is the time vector with all components squared.) If we ignore the third component of $\boldsymbol{\theta_p^T}$, then this model is similar to dead reckoning on track data. Using Eq. 8, a closed-form solution to the behavior parameter that minimizes Eq. 5 is

$$g_p(\boldsymbol{X}) = (S^T S)^{-1} S\boldsymbol{X} = \hat{\boldsymbol{\theta}}_{\boldsymbol{p}}. \tag{9}$$

Consider the example of an aircraft traveling northeast for a period of time and then abruptly changing its trajectory to travel at a slower speed and direction toward east-northeast, as shown in the left plot in Figure 2. To demonstrate SeqSAC's ability to identify different types of anomalies, we inject a spurious signal at 11 km, 3 km (east, north) that would represent a signal degradation or adversarial signal exploitation. The goal in our proposed method is to identify which tracks are anomalies, as in the plot on the right in Figure 2. Here the spurious signal and positions associated with the aircraft traveling at a different speed and heading are labeled as outliers.

Designing a behavior anomaly detector for the example in Figure 2 may be simple at first glance, but using existing methods is intractable when the detector must be robust to change. Suppose one builds an estimator that detects when the pilot is banking right given the data. To trigger an anomaly, a 30° turn for example, what would the threshold be for the banking rate—10°? What if we now need to detect when an aircraft begins to bank left? Because SeqSAC is robust to noise in data and flexible in its definition of behavior models, it is able to detect all classes of behavioral changes. The key to accomplishing this robustness and flexibility is that our approach assumes that the majority of the data observed are considered "normal;" thus, SeqSAC learns a normal behavior as more data arrive over time.

In Figure 3, we demonstrate the sequential process of SeqSAC as it builds a model over time (nonrecursive method). The figure shows observations of SeqSAC at four different iterations, where the plots in panels a and d are the first and last iterations, respectively. On the first iteration in Figure 3a, SeqSAC uses a few sequential tracks and estimates a model (green line). All tracks within an ϵ-ball (500 m for this example) about the green line are considered inliers; here it finds 90 inliers. For the next iteration, another sample set is selected in sequential order with 50% overlap. In Figure 3b, the iteration is on a sample set that overlaps the spurious signal. Here the model fits only 28 samples with the generous 500-m ϵ-ball. Figure 3c shows the moment of the turn and velocity change, and the estimated model has

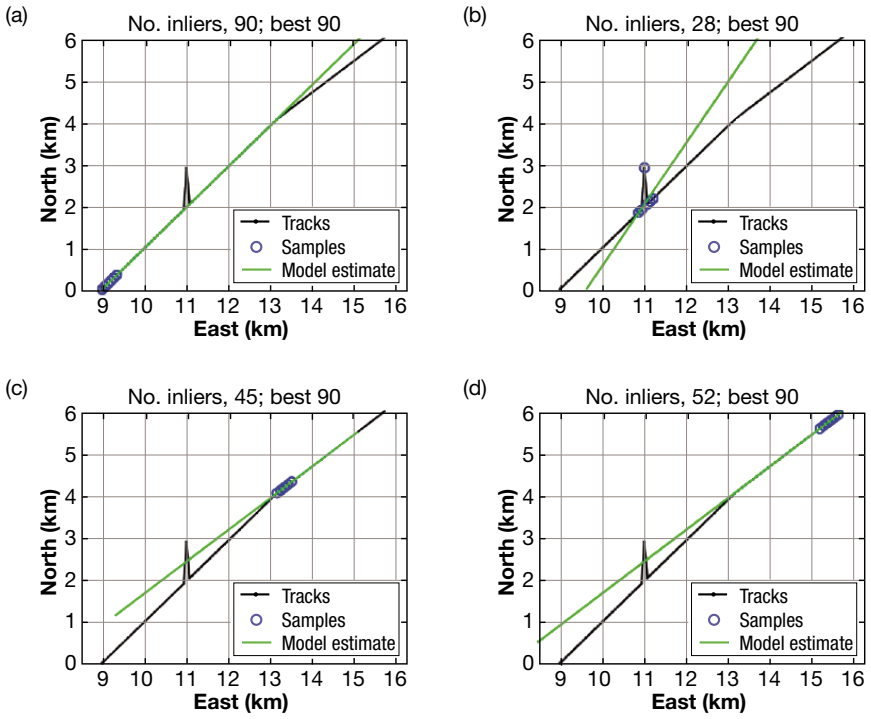

Figure 3. Demonstration of SeqSAC. (a) First iteration where 90 inliers are found with the model. (b) Sample set on the spurious signal with 28 inliers. (c) Iteration where the samples are at the turn. (d) Last iteration where 52 inliers are found with this sample set.

45 inliers. Finally, the last iteration, shown in Figure 3d, ends with 52 inliers. After sampling all the data, the best model in this example was determined to be at the first iteration with 90 inliers. All inliers, $\boldsymbol{w}$, are then used to refine the model estimate and determine the final inlier–outlier set. The outliers for this example are indicated with a red icon in the plot on the right of Figure 2.

Although we use a polynomial behavior model $f_{\boldsymbol{\theta_p}}$ and behavior estimator g_p for the development of SeqSAC in this section, we are not limited to only this model. We demonstrate in the next sections how we can extend SeqSAC to use more complex ground-path models, dynamic (guidance and control) models, and a nonlinear model.

SENSITIVITY ANALYSIS WITH DYNAMIC MODEL

We now use a dynamic model for an aircraft body using state and control parameters to observe the sensitivity of the SeqSAC algorithm when there is noise in the data. The state of the vehicle is time and position, $\boldsymbol{X}$, just as in previous sections. Although it is common to include the heading of the vehicle, ψ, in the aircraft state vector, for this article we assume the measured state of the heading is unknown; we only incorporate heading change (rate of turn) when estimating the behavior model. The control parameters that are represented by θ_d are the aircraft's ground speed, v; heading, ψ; rate of turn, $\dot{\psi}$; and altitude velocity, $\dot{z}$. More compactly, $\theta_d = (v, \psi, \dot{\psi}, \dot{z})^T$. A nonlinear solver is used in Eq. 5 to estimate the aircraft model, $\hat{\theta}_d$.

We simulated 100 trajectories with varying ground speeds between 70 and 200 m/s, altitude velocities between –100 and 100 m/s, headings in any direction, and heading rates between 0°/s and 5°/s. All trajectories start from the same location. With a probability of 50%, we assign an anomaly to a trajectory with the anomaly times between 55 and 75 samples. For this simulation, 1 sample = 1 s. If a trajectory is to have a change in behavior, we randomly select the behavior change to be (uniformly distributed) a deviation of ±20 m/s in ground speed, ±3°/s rate of turn, and ±10 m/s altitude velocity. It is possible that some trajectories have a behavior change but the change is almost miniscule. Figure 4 is an overlay of all the trajectory prototypes used for the Monte Carlo simulation. Each true anomaly location is indicated with a red square.

For each trajectory, we add Gaussian noise with zero mean and variances from 0 to 16 m^2 with 2 m^2 intervals. Thus, we have 100 trajectories with no noise but a total (including those with noise) of 900 trajectories. For each trajectory, we apply the SeqSAC algorithm to estimate

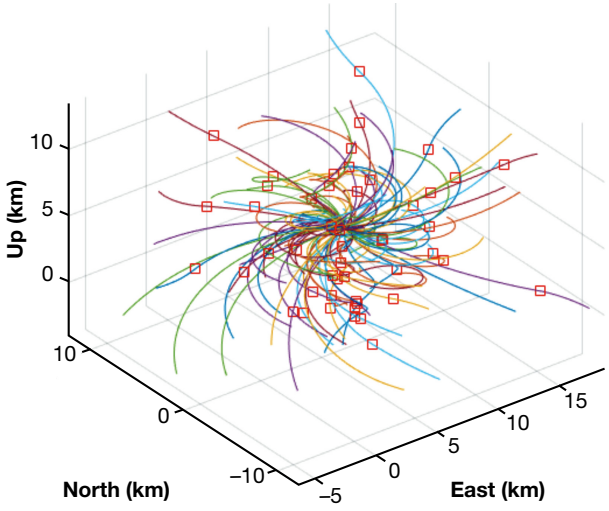

Figure 4. Overlay of all 100 trajectories used for the Monte Carlo simulations. Trajectories randomly selected to have an anomaly, as well as the location of start of the anomaly, are indicated with red squares.

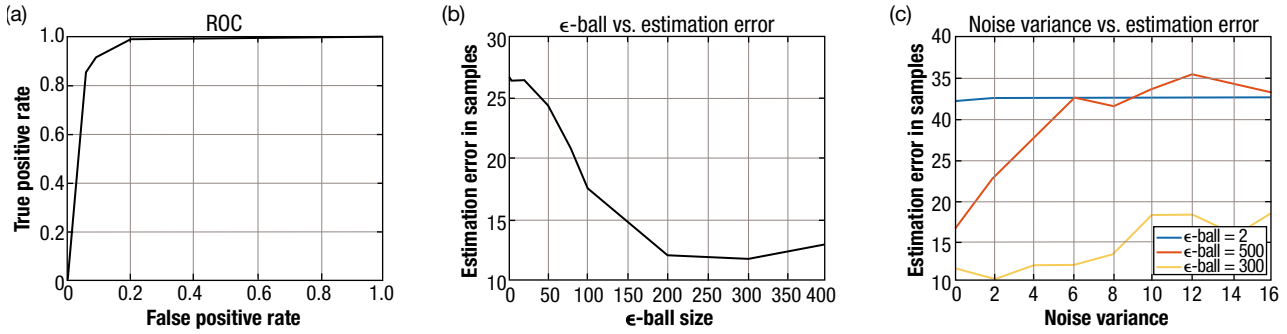

Figure 5. Results from Monte Carlo simulations. (a) ROC curve with an area under the curve of 0.932. (b) SeqSAC anomaly estimate distance error (in samples) for each ϵ-ball trial. (c) SeqSAC estimation error with increasing sample noise variance.

outliers. To demonstrate receiver operating characteristic (ROC) curves, we change the SeqSAC parameter, ϵ-ball, which is the minimum distance a data point must be from the model to be considered an inlier. (See algorithm details at the end of this article and Ref. 2). The 11 ϵ-ball values used for the simulations are 0, 2, 4, 10, 20, 50, 80, 100, 200, 300, and 400 m. (These values were only selected to support the analysis; any values could be chosen for ϵ-ball.) For the 900 trajectories, SeqSAC was used to estimate outliers with the 11 ϵ-ball values. In total, SeqSAC was called 9,900 times. The results from all the trials are shown in Figure 5.

For each simulation, the SeqSAC algorithm attempts to find anomalies in the data under the constraint of the ϵ-ball parameter. Just under half the trajectories have an anomalous behavior simulated with a change in a random combination of ground speed, rate of turn, and elevation speed vector. The SeqSAC algorithm returns a vector of inliers $\boldsymbol{w}$, for which in turn the outliers, $\overline{\boldsymbol{w}}$, are found. If the outlier vector is not empty, then it is recorded as an anomaly detection. Under the construction of this Monte Carlo simulation, SeqSAC can be viewed as a binary classifier with a trajectory either having or not having an anomaly. With a true positive meaning that the SeqSAC algorithm successfully classifies an anomaly is present and a false positive meaning that the SeqSAC algorithm incorrectly classifies a trajectory having an anomaly, we generate a plot of the false positive rate versus the true positive rate (ROC curve) in Figure 5a. The area under the ROC curve is 0.932. This demonstrates that SeqSAC has high precision and high recall under this simulation environment.

SeqSAC not only classifies whether an anomaly is present but also estimates when the first anomaly occurs in the samples. For every trajectory, we measure the absolute difference in time (samples) between the true anomaly time, w_0, and the estimated time from SeqSAC, $\hat{w}_0$. If SeqSAC does not find an anomaly, then the sample time is set to zero, which is the same time used for trajectories that, in truth, do not have an anomaly. At each ϵ-ball parameter, the estimation error, $|w_o - \hat{w}_0|$, is averaged over all trials, shown in Figure 5b. With small ϵ-ball values (less than 100), the estimation is high, which reflects how selective SeqSAC is in determining inliers (i.e., consensus must be satisfied). As the ϵ-ball value is increased, more samples are included in a consensus measurement, which in turn provides SeqSAC the flexibility to more accurately find and classify *both* inliers and outliers. As ϵ-ball increases, the estimation error decreases. However, the error does not asymptotically decrease as ϵ-ball is increased because more and more samples are included in a consensus; eventually overclassification occurs and all samples are deemed to be in consensus. We see this occur in Figure 5b where the error increases from ϵ-ball = 350 to 400. Figure 5b demonstrates that SeqSAC can be robust to noise when a large enough value for ϵ-ball is selected. In Figure 5c, the absolute difference in plots of anomaly sample times for three ϵ-ball reflects a similar story as Figure 5b. With ϵ-ball unreasonably low at 2 m, the error is flat at 33 samples. As ϵ-ball is increased, the errors decrease. As expected, there is an upward trend in error as the sample noise variance increases.

SeqSAC WITH RECURSION

The results described in the previous sections were all achieved by using SeqSAC without the recursion functionality. Next we briefly demonstrate the utility of recursive SeqSAC. With the ability to recursively search for anomalies in a data sequence, $\mathbf{X}$, a chain of behaviors can be estimated over time. In this example, a chain of anomalies and behaviors are built with Eq. 7 using the dynamic model from the previous section, f_{θ_d} and g_d. We use two different data sets. The first we generate by simulating an aircraft that begins with a left turn, proceeds to fly straight, and then makes another left turn, decreasing elevation and flying down. SeqSAC is able to detect and estimate these behavior changes, as indicated by the colored segments in Figure 6a. Then, using data from NASA, we run a simulation of an aircraft taking off, climbing to an altitude of 10 km, and descending to a landing.[4] Again, all the behavior changes are detected

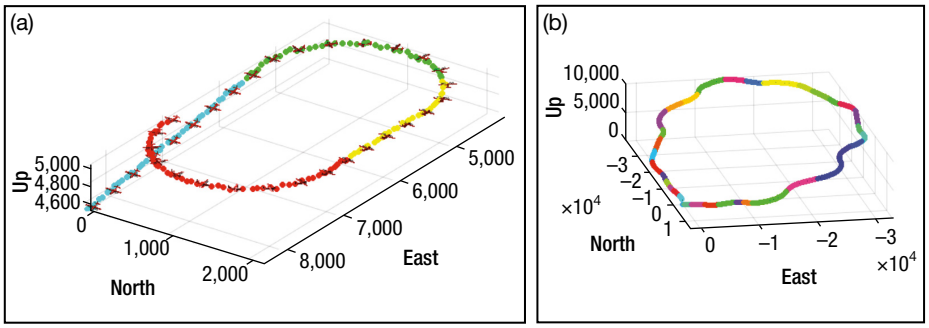

Figure 6. Segmented behaviors using recursive SeqSAC in Eq. 7. All colored data points are behaviors classified by recursive SeqSAC. (a) Segmentation from flight simulation with four distinct behavior changes. (b) Segmentation from simulated takeoff and landing data from NASA.[4]

by recursive SeqSAC and demonstrated by the different colored segments in Figure 6b.

HIGHER-ORDER MODELING WITH FINITE-STATE MACHINES

We have demonstrated that given a behavior model $f_{\boldsymbol{\theta_n}}(\cdot)$, and companion estimator, $g_n(\cdot)$, SeqSAC can accurately locate behavior anomalies with a reasonable selection of ϵ-ball. SeqSAC is flexible enough to support any model or even suite of models used. Here we briefly explore the concept of modeling and estimating a behavior with a higher-level behavior model using Advanced Framework for Simulation, Integration, and Modeling (AFSIM). Although we do not show SeqSAC detecting anomalies, we demonstrate the possible extension of SeqSAC to finite-state machine (FSM) modeling.

In this example, an entity model is built using an FSM. The entity FSM is designed to fly toward a target until it reaches a distance of 60 km and then turn away from the target. The entity will then turn toward the target again when it reaches 120 km from the target, and the cycle repeats. This effectively is a "racetrack" behavior but is dependent on higher-level parameters, θ_a: target position and speed, minimum and maximum ranges from target, and turning radius. Given only entity position data, $\boldsymbol{X}$, we estimate the FSM parameter vector, θ_a, using a nonlinear search method[5] to find the values that best choose a model that matches the data. Because the search space could be infinite and the estimation intractable, the search is limited to known constraints with reasonable speed values, turn radius, and maximum and minimum ranges from the target.

The results of the experiment with FSM behavior estimation are depicted in Figure 7. The top view of the entity in Figure 7a shows that the entity begins with flying away from the target (blue circle) then turns at max range from the target to its left with a 2-km radius until it is heading toward the target. When the entity reaches a minimum range from the target, it turns left again away from the target. The position versus time plot of this behavior is depicted in Figure 7b. What we also show, with black circles, are the estimated positions of the entity using only the estimated parameter, $\hat{\theta}_p$, which matches well with the true data.

The estimated model in Figure 7 demonstrates an important capability. Given only positions of an entity over time, it is possible to find what their behavior is at a high level via FSM. In this example, all data points were in consensus with the model, even when the entity made multiple turns. A higher-level model, like this FSM, enables SeqSAC to identify complicated anomalous behavior. In future work—with FSMs and other complex models[2]—models can be trained to support anomaly detection with SeqSAC.

DISCUSSION AND CONCLUSION

In this article, we introduced a new behavior anomaly detection algorithm, SeqSAC, which is based on the RANSAC[2] paradigm. SeqSAC is designed to estimate a vector of inliers, $\boldsymbol{w}$ (and outliers $\overline{\boldsymbol{w}}$) in data, $\boldsymbol{X}$, given a model f_{θ_n} and model estimator g_n. With this sequential

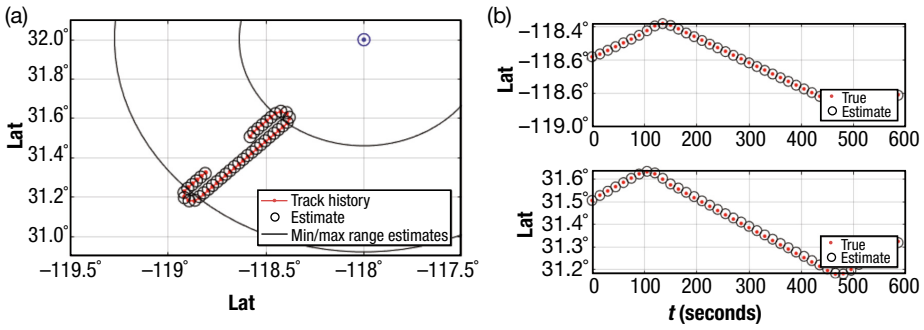

Figure 7. FSM parameter estimation with AFSIM. (a) The red line is the true entity positions. The blue circle is the target. The black circles are the positions from the estimated FSM parameter. Black arcs are the estimated minimum and maximum ranges. (b) Position vs. time view of the true and estimated values.

construct, anomalies due to noise or behavior changes can be detected (Figure 3). We demonstrated in Monte Carlo simulations (Figure 5) that SeqSAC is accurate in classifying anomalies in data as well as in identifying the sample time of the first anomaly. We also showed how a complex flight trajectory can be segmented into multiple behaviors using the recursive SeqSAC algorithm in Eq. 7.

We demonstrated SeqSAC working with a polynomial behavior model, f_{θ_p}, and a dynamic model, f_{θ_d}, and work is ongoing to extend its application to expected and recognized flight patterns. We explored possibly extending SeqSAC to FSM modeling using AFSIM and an entity "pursue" and "evade" behavior. We demonstrated the ability to estimate an entity behavior, g_a, given an entity FSM model f_{θ_a}. Although we did not demonstrate the performance of SeqSAC, the same model and estimator can be used in an FSM framework to be demonstrated in future work.

The amount of time for SeqSAC to process data and find anomalies depends on the models it uses. To construct the behavior anomaly detection with SeqSAC, a set of models and model parameter estimators are needed. Without recursion, SeqSAC is $O(N)$ complex with N samples. The complexity is higher with recursion but is data dependent. More recursion steps are needed if an entity changes its behavior frequently over the N samples, and vice versa. The processing time of SeqSAC is also dependent on the complexity of the model estimators, g_n. For each sample, a call to g_n is made. For example, the FSM estimation used in the previous section required over a minute to estimate g_a, whereas the model estimator for the dynamic model, g_d (see the sections on sensitivity analysis and SeqSAC with recursion), only required seconds to process all the data samples—even with recursion.

The accuracy—in time—in finding anomalies by SeqSAC depends on the time resolution in the data samples and SeqSAC design parameters. If the samples are separated by 5 min, then SeqSAC can only estimate within 5 min when an anomaly occurs. The more subtle the behavior change, the less accurate SeqSAC can be. This can be accounted for by selecting smaller ϵ-ball values at the risk of increasing the false positive rate.

Recursive SeqSAC is powerful in building a chain of simple behaviors to represent a complex behavior, but there are limitations. We have observed that without a properly designed exit criteria, the recursion could fall into an infinite loop, continually leaving a few data points as outliers. This infinite recursion is a trivial problem to solve but must be accounted for in designing SeqSAC.

Recursive SeqSAC enables multiple applications in incorporating flight plans, coordinated behavior detection, and compression and decompression (CODEC) of data. As shown in Figure 6b, we used SeqSAC to find multiple behavior changes in a complete flight (takeoff and landing). In this example, each small turn was treated as an anomaly, resulting in several anomalies. Although several anomalies were detected, this chained sequence of behaviors (from the anomalies) via recursive SeqSAC is a flight plan learning mechanism. Thus, the number of anomalies may be reduced by feeding SeqSAC with the learned flight plan model. More generally, a flight plan can be fed into SeqSAC as a model for identifying flight plan deviations. If the pilot deviates from the flight path, whether by position or speed or a combination, anomalies arise. Finally, incorporating the impact of wind in the models can also reduce the number of false anomaly detections and yet identify anomalous behavior in flight patterns. Extended application of SeqSAC to identify coordinated behavior changes among many entities is an intriguing prospect. The high precision and recall afforded by SeqSAC supports detection of behavior changes that occur simultaneously across multiple entities, potentially indicating coordination. (Reference the 100-entity Monte Carlo simulation depicted in Figure 4.) Additionally, with the method of applying SeqSAC recursively to segment a trajectory that spans several samples, SeqSAC affords the ability to compress and decode location histories of entities.

ACKNOWLEDGMENTS: Many thanks to Debora Arena, Dr. Michael Hassien, Dr. Bryan Herdlick, and Justin Shoger for the guidance, thought-provoking questions, and reviews during the development of this article.

REFERENCES

[1]V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, https://doi.org/10.1145/1541880.1541882.

[2]V. M. Janakiraman and D. Nielsen, "Anomaly detection in aviation data using extreme learning machines," in *Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, Jul. 24–29, 2016, pp. 1993–2000, https://doi.org/10.1109/IJCNN.2016.7727444.

[3]M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, https://doi.org/10.1145/358669.358692.

[4]"FLTz flight simulator," data set, NASA ARC, Jan. 2011, https://c3.nasa.gov/dashlink/projects/42/.

[5]J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, no. 1, pp. 112–147, 1998, https://doi.org/10.1137/S1052623496303470.

SeqSAC Algorithm

procedure $SEQSAC(\boldsymbol{X}, f, g, \boldsymbol{\theta}_S, \boldsymbol{w}_0)$
Input:
$\boldsymbol{X}$: Matrix of entity track history
$f()$: Model function from Eq. (2)
$g()$: Estimator function from Eq. (4)
$\boldsymbol{\theta}_S = \{N_{\max}, N_{\min}, \epsilon_{\text{ball}}\}$: SEQSAC parameters
$N_{\max}$: Max Iterations
$N_{\min}$: Minimum number of samples to use to estimate model
ϵ_{ball} : Max distance sample must be from model estimate to be classified as an inlier

Output:
w: Weight vector for $\boldsymbol{X}$ classified as inliers
$\widehat{\boldsymbol{\theta}}$: Model parameter estimate

idx ← 0 # Current sample index
$K \leftarrow 0$ # *Number of samples in consensus*
inliers ← {**0**} # *Vector of sample indexes classified as inliers,*
initialized as Null set
$\boldsymbol{t} \leftarrow \boldsymbol{X}$ # *Extract all time stamps from entity history*
$\boldsymbol{x} \leftarrow \boldsymbol{X}$ # *Extract all positions from entity history*
while $k < N_{\max}$ **do**
# *Extract* N_{min} *sub-samples*
$\boldsymbol{t}_k, \boldsymbol{x}_k \leftarrow \boldsymbol{X}(\text{idx} : \text{idx} + N_{\min})$
$\boldsymbol{\theta}_{k,f} \leftarrow g(\boldsymbol{x}_k)$ # *Estimate with* N_{min} *sub-samples*
Use all timestamps from input for predictions
$\boldsymbol{x}_{\text{p}} \leftarrow f(\boldsymbol{t}, \boldsymbol{\theta}_{k,f})$
Distance between i-th predicted and input samples
$\boldsymbol{d}$: $d_i \leftarrow \left\| \boldsymbol{x}_i - \boldsymbol{x}_{\text{p},i} \right\|_2^2 \; \forall \; i \in 0 \ldots N$
Classify all samples within ϵ_{ball} *as the inlier set*
$\mathbf{j} \leftarrow i \; \forall \; d_i < \epsilon_{\text{ball}}$
Consensus measure
$\Omega_{\mathbf{j}} \leftarrow (length\ of\ \mathbf{j}\ vector)$
if $\Omega_{\mathbf{j}} < K$ # *If we have a better consensus*
$K \leftarrow \Omega_{\mathbf{j}}$
Update behavior estimate using only inlier data
$\widehat{\theta_f} \leftarrow g(\boldsymbol{X}(\boldsymbol{j}))$
$\mathbf{w} \leftarrow \mathbf{j}$
end if
# *Move the sample index in time*
idx ← idx + round $\left(\frac{N_{min}}{2}\right)$ # *50% overlap over samples*
end while

return $\mathbf{w}, \hat{\theta}_f$

end procedure

Kristofor B. Gibson, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Kristofor B. Gibson supervises APL's System Concept Development Section and is the technical lead for the Tactical Decision Aid (TDA) project. He earned a BSEE from Purdue University and an MSEE and PhD with a focus on signal and image processing from the University of California San Diego. Dr. Gibson leads a team developing concepts, algorithms, and software to support the Navy in building decision aids for warfighters to help them better employ the expanding set of complex capabilities and systems at their disposal. He previously worked as a principal investigator for research and development applications in support of maritime domain awareness at Naval Information Warfare Center Pacific for 15 years. His current research interests are in signal and image processing, computer vision, atmospheric propagation, decision-making, machine learning, and artificial intelligence. He developed SeqSAC and Recursive SeqSAC in support of the TDA project at APL. He is a recipient of the 2010 AFCEA International & US Naval Institute Copernicus Award and the United States Intelligence Community National Intelligence Meritorious Unit Citation (2012), holds three patents, and has authored over 20 publications. His email address is kristofor.gibson@jhuapl.edu.