A New Clustering Algorithm for Processing GPS-Based Road Anomaly Reports With a Mahalanobis Distance

Zhaojian Li, *Student Member, IEEE*, Dimitar P. Filev, *Fellow, IEEE*, Ilya Kolmanovsky, *Fellow, IEEE*, Ella Atkins, *Senior Member, IEEE*, and Jianbo Lu, *Senior Member, IEEE*

***Abstract*— This paper considers a new clustering algorithm for processing time-evolving road anomaly reports. Two cluster categories, main and outlier, are defined to deal with outliers as well as to capture the evolving nature of road anomalies. The Mahalanobis distance is exploited to quantify the similarity between a new report and the existing clusters. The clusters are maintained online and the Woodbury matrix inverse lemma is used for their recursive updates. The proposed clustering algorithm can localize isolated anomalies and compress information for densely distributed anomalies. A simulation is presented to demonstrate the efficacy of the proposed algorithm.**

***Index Terms*— Evolving clustering algorithm, Mahalanobis distance, road anomaly report, Woodbury matrix inversion lemma.**

I. Introduction

Mobile sensing and data sharing offer new opportunities to advance intelligent transportation systems. Modern vehicles are equipped with sophisticated sensors and control units that can be exploited to obtain road and environmental information in real time. References [1]–[3] provide examples of traffic density estimation, road friction coefficient estimation and pothole detection, respectively. Sensed information can be sent to a server, *e.g.,* the cloud, to be further processed, crowd-sourced, then shared with other vehicles and road agencies.

Road anomalies such as potholes or bumps are annoying events that can cause ride discomfort and vehicle damage. If available, anomaly maps can be used to enhance route planning, improve suspension control [4], [5] and inform road maintenance activities. Anomaly detection algorithms have been developed in previous work. For example, a pothole detector with three external accelerometers was developed using machine learning techniques in [3]. In [6] and [7], we developed a road anomaly detection algorithm based on a half-car model by exploiting a multi-input observer. Promising detection performance was demonstrated in a test vehicle using standard sensors.

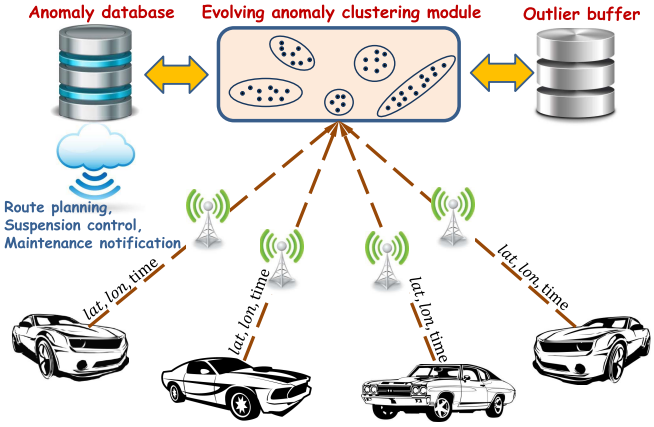


Fig. 1. Vehicle-to-Cloud-to-Vehicle anomaly detection and information sharing.

Vehicles that perform road anomaly detection can be integrated into a Vehicle-to-Cloud-to-Vehicle framework as illustrated in Figure 1. Such vehicles used as mobile sensors can be either special vehicles or customer-owned vehicles who choose to participate in the program. Once anomalies are detected, anomaly locations, *e.g.*, from Global Positioning System (GPS) coordinates, are sent to the cloud, where a clustering module is implemented to process raw anomaly reports. Clusters with high credibility score are stored in a cloud database where their locations can later be broadcast to other vehicles and road agencies. Clusters with low credibility score are stored in a buffer and not shared. In this paper, we develop a novel clustering algorithm that can process raw reports and retrieve useful anomaly information. The desired clustering algorithm has the following properties:

- *No assumptions on the number of clusters*. The number of road anomalies may not be known in advance and is continuously evolving. New anomalies can develop and old anomalies can disappear once repaired. The algorithm hence should not assume a constant number of clusters [8], [9].
- *Ability to handle outliers*. False alarms can sometimes occur. The clustering algorithm should be able to discriminate outliers and not broadcast outlier information to vehicles and road agencies.

Manuscript received January 19, 2016; revised May 21, 2016 and August 3, 2016; accepted September 23, 2016. Date of publication October 12, 2016; date of current version June 26, 2017. This work was supported in part by Ford Motor Company and in part by University of Michigan Alliance. The Associate Editor for this paper was J. M. Alvarez.

Z. Li, I. Kolmanovsky, and E. Atkins are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48105 USA (e-mail: zhaojli@umich.edu; ilya@umich.edu; ematkins@umich.edu).

D. P. Filev and J. Lu are with the Research and Advanced Engineering, Ford Motor Company, Dearborn, MI 48121 USA (e-mail: jlu10@ford.com; dfilev@ford.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2016.2614350

- *Consideration of anomaly evolution*. Road anomalies are evolving, that is, new potholes may occur and old potholes may be fixed. The clustering algorithm must be able to deal with change in aggregated reports.
- *Localization of isolated anomalies and information compression for stretched (densely spaced) anomaly segments*. The algorithm should also be able to accurately localize isolated anomalies and, from the perspective of road information sharing, it is desirable to aggregate information from a segment with densely spaced multiple anomalies.
- *Memory and computation efficiency*. We envision a fleet of vehicles that are equipped with anomaly detectors (e.g., the ones developed in [6]) travelling around to enable sufficient coverage. Therefore, the clustering algorithm needs to process large-scale data streams as efficiently as possible. Cluster information should be stored in a compact data structure and updated with minimal computational overhead.

In general, clustering algorithms partition data into groups based on underlying patterns. These algorithms are widely applied in the fields of image processing [10], data mining [11], and diagnostics and prognostics [12]. Many clustering algorithms are designed to deal with static data [8], [13], [14], that is, cases where all data are available in advance. These algorithms are not applicable to processing anomaly reports since the reports are dynamic and time-evolving. Recently, clustering algorithms have been developed to deal with evolving data streams. The CluStream algorithm [9] exploits micro-clusters to summarize information for a set of data points. The micro-clusters are updated online with new stream inputs and a weighted k-means algorithm is applied offline on the micro-clusters to obtain the final clusters. While good accuracy can be obtained, the algorithm assumes a constant number of clusters so it cannot be used in our problem. In [15], a streaming k-means clustering algorithm is developed with a divide-and-conquer strategy. It optimizes a k-means objective function and can generate more than k clusters. However, the obtained clusters are hypercircles which cannot be used to compress information for stretched anomaly segments. Also, this approach does not easily handle outliers and the evolving nature of anomalies.

An extended Gustafson-Kessel algorithm is developed in [16], where Mahalanobis distance is exploited to measure the similarity between clusters and new data points. The cluster center and covariance matrix are updated recursively with new data inputs. Updated clusters are hyperellipsoids with arbitrary orientation. The algorithm is applicable to real-time pattern recognition and information compression. However, the algorithm in [16] is only applicable to spatial data and cannot handle directly dynamic data with temporal features such as road anomaly reports. Also, the algorithm in [16] is not able to deal with outliers and cannot capture the road anomalies that change over time.

In this paper, we develop a novel clustering framework that satisfies all specified requirements. We exploit Mahalanobis distance as the similarity metric. Two cluster types, the outlier cluster and the main cluster, are defined based on their computed credibility values. The credibility values are reflected in accumulated reports, that is, cluster credibility grows with the increased number of reports. A decaying function is used to discount the credibility with time to deal with situations that road anomalies disappear due to repair. A cluster feature vector is defined by a weight, center, covariance matrix inverse, creation time and a label. Clusters are updated in a single-pass setting. A Woodbury matrix inversion lemma [17] is exploited to simplify the covariance matrix update and avoid possible singularity issues in numerical computations. Clusters are pruned based on their weights and creation time to deal with outliers as well as anomaly changes over time. Memory and computations are light and simulation results demonstrate the efficiency of the proposed clustering algorithm.

The paper is organized as follows. Section II presents background on Mahalanobis distance and its relation to the χ^2 distribution. Section III is devoted to the discussion of cluster feature definition and the clustering algorithm. Simulation results are described in Section IV, followed by a summary and discussion in Section V.

II. Background

A. Mahalanobis Distance and χ^2 Distribution

The Mahalanobis distance measures the similarity between a point and a cluster of points [18]. It generalizes a notion of number of standard deviations between a point and the mean of the cluster for multi-dimensional data. The distance grows as the point moves away from the mean along each principal component axis. As a result, the distance is unitless and scale-invariant, and accounts for the distribution and correlations of the cluster data. Let $x \in \mathbb{R}^n$ be a data point. Let μ and Σ be the mean and covariance matrix of a cluster of points denoted by $\mathcal{C}$, respectively. The Mahalanobis distance between x and $\mathcal{C}$, $\mathcal{D}(x, \mathcal{C})$, is defined as:

$$\mathcal{D}(x, \mathcal{C}) = \sqrt{(x-\mu)^{\mathrm{T}} \Sigma^{-1} (x-\mu)}. \tag{1}$$

Note that Mahalanobis distance in (1) coincides with the Euclidean distance between x and μ in a special case with Σ being the identity matrix.

Suppose the data points are normally distributed around the cluster center μ with covariance Σ, *i.e.*, $X \sim \mathcal{N}_n(\mu, \Sigma)$, where $\mathcal{N}_n$ represents the multivariate normal distribution of dimension n. Define

$$Z = \Sigma^{-\frac{1}{2}}(X-\mu) = [Z_1,\ Z_2, \cdots,\ Z_n]^{\mathrm{T}}.$$

It is straightforward to show that $Z \sim \mathcal{N}_n(\mathbf{0}_n, \mathbf{I}_n)$, where $\mathbf{0}_n$ and $\mathbf{I}_n$ represent the zero vector of dimension n and identity matrix of dimension n, respectively. As a result, the Mahalanobis distance in (1) between X and $\mathcal{C}$ is:

$$\mathcal{D}^2(X, \mathcal{C}) = Z^{\mathrm{T}} Z = Z_1^2 + Z_2^2 + \cdots + Z_n^2, \tag{2}$$

which means that $\mathcal{D}^2(X, \mathcal{C})$ is chi-square distributed with degrees of freedom n, *i.e.*, $\mathcal{D}^2(X, \mathcal{C}) \sim \chi_n^2$. The chi-square value is often associated with a p-value, which is defined as the probability of obtaining a result equal to or "more extreme" than what is observed. The chi-square distribution

TABLE I

CROSS-REFERENCE TABLE OF p-VALUE, CONFIDENCE INTERVAL AND SIGMA BAND FOR $n = 1$, AND χ^2 VALUES FOR $n = 1, 2, 3$

σ band	1σ	2σ	3σ	4σ
confidence interval (%)	68.3%	95.45%	99.73%	99.99%
p-value	0.317	0.0455	0.0027	0.000006
$\chi^2_1(p)$	1	4	9	16
$\chi^2_2(p)$	2.3	6.18	11.83	19.33
$\chi^2_3(p)$	3.53	8.02	14.16	22.06

is frequently used in statistical hypothesis testing. The value $(1-p)$ is known as the confidence interval that represents the probability of $\mathcal{D}^2(X, \mathcal{C}) < \chi^2_n(p)$. The cross references of the p-value, confidence interval and the sigma values (σ, standard deviation) for $n = 1$ and χ^2_n values for some low dimensions are given in Table I.

B. Preliminaries

In this subsection, we introduce the following lemma that will be used in subsequent developments.

Lemma 1 [17]: Let A, B, C, D be matrices of appropriate dimensions. Suppose matrices A, C, and $C^{-1} + DA^{-1}B$ are nonsingular, then

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}. \tag{3}$$

Lemma 1 is often referred to as the *Woodbury matrix inversion lemma*.

III. ANOMALY REPORT STREAM CLUSTERING ALGORITHM

In this section, we develop a new clustering algorithm to process road anomaly report streams that satisfies all the desired properties specified in Section I. We first introduce a notion of cluster features, followed by the detailed description of our clustering algorithm.

A. Cluster Features

The main goal of our clustering algorithm is to obtain anomaly information by processing aggregated anomaly reports from vehicles. To achieve this goal, we represent each cluster $\mathcal{C}_i$, $i = 1, 2, \cdots, c$, with a tuple,

$$\mathcal{C}_i = (w_i, v_i, \Sigma_i^{-1}, t_i^0, \mathcal{L}_i), \tag{4}$$

where $w_i = \sum_{k=1}^{M_i} f(t - t_{ik})$ is the weight of cluster $\mathcal{C}_i$ with M_i being the number of anomaly reports in the cluster. The time instants t and t_{ik} denote the current time instant and the time instant that x_{ik}, the kth report in cluster i, was merged to cluster i, respectively. Time stamps t and t_{ik} have a common time unit, for example, in days. The function $f(\tau)$ is a decreasing function of elapsed time to discount cluster weights. In this paper, we use $f(\tau) = \alpha^{-\lambda\tau}$ where $\alpha > 1$ and $\lambda > 0$ are two positive scalars. Specifically, based on our numerical experiments, we recommend $\alpha = 2$. The decaying functions with different λ's are illustrated in Figure 2.

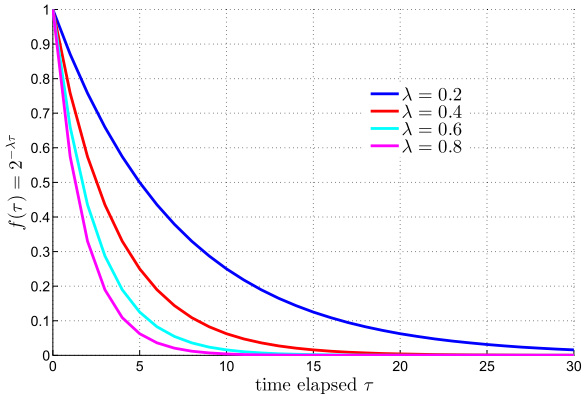


Fig. 2. Decaying function $f(\tau)$ used to discount cluster weights based on elapsed time for different λ's.

The weight w_i reflects the credibility score of the cluster where high w_i corresponds to high credibility. Note that the weight of a newly received report is one and the weight decays as a function of the elapsed time.

The cluster center v_i is defined as a weighted mean:

$$v_i = \frac{\sum_{k=1}^{M_i} f(t - t_{ik}) x_{ik}}{\sum_{k=1}^{M_i} f(t - t_{ik})}, \tag{5}$$

and Σ_i is the weighted covariance matrix of the cluster defined as

$$\Sigma_i = \frac{\sum_{k=1}^{M_i} f(t - t_{ik})(x_{ik} - v_i)(x_{ik} - v_i)^{\mathrm{T}}}{\sum_{k=1}^{M_i} f(t - t_{ik})}. \tag{6}$$

We track the inverse of Σ_i, instead of Σ_i, for the convenience of recursive computations as detailed in Section III-C. The time stamp $t_i^0 > 0$ represents the time instant when cluster $\mathcal{C}_i$ is first created.

The variable $\mathcal{L}_i \in \{m, o\}$ in (4) serves as a label indicating the cluster type. We specify two cluster types, main clusters (m-clusters, $\mathcal{L}_i = m$) and outlier clusters (o-clusters, $\mathcal{L}_i = o$). The m-clusters are the clusters with high credibility that are believed to represent true anomalies. The credibilities of clusters are reflected in the cluster weight w_i, where high w_i implies high credibility. On the other hand, o-clusters represent outliers due to false alarms or those representing true anomalies that do not yet have high aggregated weights.

We note that the m-clusters and o-clusters can be interchanged as weights change. This allows new anomalies to become m-clusters and removed (repaired) anomalies to become outliers. Two thresholds h_m and h_o, $h_o > h_m > 0$, are introduced to capture the interchange capability. For an m-cluster, if few reports are obtained and the cluster weight decays such that $w_i < h_m$, then cluster i will be relabeled an o-cluster. For an o-cluster, if the cluster weight, with aggregated reports, increases such that $w_i > h_o$ then the cluster is relabeled as an m-cluster. The constraint $h_m < h_o \cdot \alpha^{-\lambda}$ avoids clusters repeatedly switching labels. The thresholds h_m and h_o can be set as a function of annual average daily traffic for each road segment.

Fig. 3. Clusters in the forms of ellipsoids.

On the other hand, if an o-cluster fails to become an m-cluster after a certain period of time T_o, it means that the o-cluster corresponds to false alarms and should be deleted from the outlier buffer. We check all the o-clusters periodically, *e.g.,* at the end of each day. If $t - t_i^0 > T_o$, we then delete $\mathcal{C}_i$ from the outlier buffer, where t is the current time and t_i^0 is the cluster creation time.

The obtained clusters are hyperellipsoids with orientation determined by the principal axes of their covariance matrices Σ_i. The cluster representation can localize true location for isolated events and can compress information for a stretch of anomaly events with arbitrary orientation as illustrated in Figure 3. A stretch of three anomalies is included in Cluster 1 with an orientation aligned with the road. Moreover, it can accurately indicate the anomaly location for an isolated anomaly as in Cluster 2.

The m-cluster information is stored in a cloud database and can be shared between vehicles for route planning, suspension control [4], or other purposes. The information can also inform road agencies to schedule maintenance activities. The o-clusters are stored in a buffer that is not shared with other users.

B. Cluster Maintenance Algorithm

In this paper, we develop an algorithm (Algorithm 1 below) to process road anomaly report streams. When a new anomaly report arrives, there are two possible scenarios. First, if the newly arrived report is "close" to some existing clusters, then the data should be merged into the "closest" one. On the other hand, if there is no existing cluster or the data is not close to any of the existing clusters, a new cluster should be created and centered at the newly reported location. The "closeness" or similarity of the newly reported location and existing clusters is measured by the Mahalanobis distance discussed in Section II-A. These scenarios (or conditions) are characterized by the *if* statements at Steps 4 and 9 in Algorithm 1.

Note that since the Mahalanobis distance is unitless and scale-invariant, we are able to directly process GPS coordinates without transforming them to state plane coordinates in Euclidean space.

Algorithm 1 Anomaly Report Stream Clustering Algorithm

Constant Parameters: p, α, λ, h_o, h_m, T_o, γ
Inputs: $\mathcal{C}$, X
Outputs: $\mathcal{C}^+$

1: *top:*
2: **do**
3: Read next report $x_k \in X$
4: **if** no cluster exists, **then**
5: Initialize the first cluster $\mathcal{C}_1$:

$$w_1 = 1,\ v_1 = x_1,\ \Sigma_1^{-1} = \gamma\,\mathbf{I}_2,\ \mathcal{L}_1 = o,\ t_1^0 = t.$$

6: **else**
7: Calculate the Mahalanobis distances to all existing clusters:

$$\mathcal{D}^2(x_k, \mathcal{C}_i) = (x_k - v_i)\Sigma_i^{-1}(x_k - v_i)^{\mathrm{T}}, \quad i = 1, \cdots, c,$$

where c is the number of clusters.
8: Find the closest cluster as:

$$i^* = \arg\min_{i=1,\cdots,c} \mathcal{D}^2(x_k, \mathcal{C}_i).$$

9: **if** $\mathcal{D}^2(x_k, \mathcal{C}_{i^*}) \le \chi_2^2(p)$, **then**
10: Update the covariance matrix inverse of $\mathcal{C}_{i^*}$ using (13) and (14).
11: Update the center of $\mathcal{C}_{i^*}$ using (10).
12: Update the weight of $\mathcal{C}_{i^*}$ using (9).
13: **if** $\mathcal{L}_{i^*} = o$ and $w_{i^*} > h_o$, **then**
14: Set $\mathcal{L}_{i^*} = m$ and update the cloud database with C_{i^*}.
15: **else**
16: Create a new cluster and increment $c = c + 1$.
17: Initialize the new cluster $\mathcal{C}_c$ as:

$$w_c = 1,\ v_c = x_k,\ \Sigma_c^{-1} = \gamma\,\mathbf{I}_2,\ t_c^0 = t,\ \mathcal{L}_c = o.$$

18: **while** More than m minutes before the end of day t
19: Within m minutes to the end of day t:
20: Update the cluster weights:

$$w_i^+ = w_i \cdot \alpha^{-\lambda}, \quad i = 1, 2, \cdots, c.$$

21: Check the m-clusters:
22: **if** $\mathcal{C}_i$ is an m-cluster and $w_i < h_m$ after the update, **then**
23: Set $\mathcal{L}_i = o$; delete it from the cloud database; send it to the outlier buffer.
24: Check the o-clusters:
25: **if** $\mathcal{C}_i$ is an o-cluster and $t - t_i^0 = T_o$, **then**
26: delete $\mathcal{C}_i$ from the outlier buffer.
27: increment the day count: $t = t + 1$.
28: **go to** *top*.

Let $\mathcal{C}$ and $\mathcal{C}^+$ define the set of old cluster features and updated cluster features, respectively. Let X define the sequence of anomaly reports. Suppose there are c existing clusters when a new report $x_{\text{new}} = (lon, lat)$ arrives. Then based on (1), the squared Mahalanobis distance between data point x and cluster $\mathcal{C}_i = (w_i, v_i, \Sigma_i^{-1}, t_i^0, \mathcal{L}_i)$,

$i = 1, 2, \cdots, c$ is calculated as

$$\mathcal{D}^2(x, \mathcal{C}_i) = (x - v_i)^{\mathrm{T}} \Sigma_i^{-1} (x - v_i). \tag{7}$$

The cluster, i^*, with the minimum distance can be obtained as

$$i^* \in \arg \min_{i=1,\cdots,c} \mathcal{D}^2(x, \mathcal{C}_i). \tag{8}$$

Finding the "closest" cluster i^* using (7) and (8) is illustrated by Steps 7 and 8 in Algorithm 1.

GPS data measurements are, in general, normally distributed [19]. With the weighted mean formulation of cluster center (5), it is typically the case that the cluster center, with aggregated reports, converges to the true anomaly location (See Figure 11 in Section IV). This justifies an assumption made in rationalizing some of the thresholds used by the proposed algorithm that the new reports are normally distributed around the cluster center. As discussed in Section II-A, suppose x is normally distributed around v_{i^*} with covariance matrix Σ_{i^*}. Then $\mathcal{D}^2(x, \mathcal{C}_{i^*}) \sim \chi_n^2$, where $n = 2$ is the dimension of x. We thus define a threshold parameter $\chi_2^2(p)$ to determine whether a data point is close enough to the cluster and can be included in the cluster. The bound can be different between urban and rural areas due to GPS accuracy characteristics. Consequently, if the squared Mahalanobis distance to the closest cluster is within the bound $\mathcal{D}^2(x, \mathcal{C}_{i^*}) \leq \chi_2^2(p)$ then we merge x into cluster i^*, which is captured by Step 9 in Algorithm 1. The weighted mean v_{i^*}, weighted covariance matrix Σ_{i^*}, and cluster weight w_{i^*} are, respectively, updated as

$$w_{i^*}^+ = w_{i^*} + 1, \tag{9}$$

$$v_{i^*}^+ = \frac{w_{i^*} v_{i^*} + x}{w_{i^*} + 1}, \tag{10}$$

$$\Sigma_{i^*}^+ = \frac{w_{i^*} \Sigma_{i^*} + (x - v_{i^*})(x - v_{i^*})^{\mathrm{T}}}{w_{i^*} + 1}, \tag{11}$$

where the superscript "+" designates updated value. The weight and mean updates are given by Steps 11 and 12 in Algorithm 1, respectively. Note that the inverse of covariance matrix update at Step 10 in Algorithm 1 does not use (11). Instead, we exploit the Woodbury Inverse Lemma to simplify the computations as will be discussed in the next subsection.

Suppose a cluster i^* is an o-cluster and after the update, the weight w_{i^*} becomes greater than threshold h_o. In this case we relabel cluster i^* as an m-cluster and its information is stored on a cloud database and shared with vehicles and agencies. This relabeling procedure is presented by Steps 13 and 14 in Algorithm 1.

On the other hand, if $\mathcal{D}^2(x, \mathcal{C}_{i^*}) > \chi_2^2(p)$, that is, the data is outside all cluster boundaries, we increment $c = c + 1$ and assign a new cluster $\mathcal{C}_c$ to x,

$$\mathcal{C}_c = (1, x, \gamma \mathbf{I}_n, o, t), \tag{12}$$

where $\gamma > 0$ is a scalar that initializes the covariance matrix inverse and t is the current time (in days). This initialization is illustrated by Steps 15-17 in Algorithm 1.

At the end of each time period, the weight of each cluster is decayed by multiplying it with $\alpha^{-\lambda}$. We check the weights of all m-clusters: if any m-cluster has a weight less than h_m, then it is removed from the database and sent to o-clusters in the outlier buffer. The o-cluster weights are also updated. If the cluster creation day of an o-cluster is less than the current day minus T_o, a time duration threshold to delete o-clusters if they fail to become m-clusters, then the cluster is removed from storage. These relabeling and pruning procedures are characterized by Steps 20-26 in Algorithm 1.

Note that anomaly reports are processed in a single pass; that is, they are all processed exactly once. Anomaly information is summarized in cluster features without storing separate reports. This reduces memory and computation resources requirements in comparison to algorithms that store individual reports.

C. Recursive Computation of Matrix Inverse

The expression (11) provides a simple way to update the covariance matrix for clusters. However, after each update, the inverse of the covariance matrix must be computed to estimate Mahalanobis distance from next arrived point as in (7). This recursive computation of matrix inverse may cause singularity issues due to numerical ill-conditioning. As an alternative, we exploit Lemma 1, the *Woodbury matrix inversion lemma*, to address this issue.

Let $A = \frac{w_i}{w_i+1}\Sigma$, $B = x - v_i$, $C = \frac{1}{w_i+1}$, and $D = (x - v_i)^{\mathrm{T}}$. From (3) and (11), it follows that

$$\begin{aligned}
(\Sigma_i^+)^{-1} &= \Big(\frac{w_i}{w_i+1}\Sigma + (x - v_i)\frac{1}{w_i+1}(x - v_i)^{\mathrm{T}}\Big)^{-1} \\
&= \frac{w_i+1}{w_i}\Sigma_i^{-1} - \frac{w_i+1}{w_i}\Sigma_i^{-1}(x - v_i)\Big[(w_i+1) \\
&\quad + (x - v_i)^{\mathrm{T}}\frac{w_i}{w_i+1}\Sigma_i^{-1}(x - v_i)\Big]^{-1} \\
&\quad \cdot (x - v_i)^{\mathrm{T}}\frac{w_i}{w_i+1}\Sigma_i^{-1} \\
&= (1 + \frac{1}{w_i})\Sigma_i^{-1}\big(\mathbf{I}_n - K_i \Sigma_i^{-1}\big),
\end{aligned} \tag{13}$$

where

$$K_i = (x - v_i)\Big[w_i + (x - v_i)^{\mathrm{T}}\Sigma_i^{-1}(x - v_i)\Big]^{-1}(x - v_i)^{\mathrm{T}}. \tag{14}$$

Note that calculation of K_i in (14) requires only a scalar inversion. Recursive matrix inversion computations with (13) replace numerical matrix inversion with a simple algebraic calculation. This technique greatly simplifies the calculation and resolves possible singularity issues. The update of the covariance matrix inverse using (13) and (14) is illustrated by Step 10 in Algorithm 1.

D. Parameter Selection Discussion

The following parameters must be specified to implement the proposed clustering algorithm.

- p-value. The p-value is required to obtain a chi-square value bound $\chi_2^2(p)$ based on Table I. This bound controls the distance over which reports can be merged to an existing cluster, affecting the cluster mean and covariance updates (10) and (11). This parameter depends on

GPS accuracy that may vary from urban to rural areas. We recommend to use $p = 0.0027$.

- α and λ in the decay function. This pair of positive parameters is used in the decay function to control dependence on old data. Large α and λ lead to less dependence on old data. We recommend to use $\alpha = 2$ and tune λ only based on data. Note that maintenance information from road agencies can be incorporated to change decay rate if we know anomalies in certain areas have been repaired.
- Thresholds $h_o > h_m > 0$. These two thresholds define the criteria for switching between an m-cluster and an o-cluster. These thresholds depend on the annual average daily traffic in each of the road segments. The higher the average daily traffic is, the higher h_o and h_m should be. The parameters h_m and h_0 also depend on the performance of the road anomaly detector, e.g., false positive (false alarm) rate and false negative (missed detection) rate.
- Time unit and pruning period T_o. Since the life cycle of road anomalies is typically at least a few days, it is reasonable to use "day" as the time unit. The time duration parameter T_o controls how long we keep an o-cluster. If an o-cluster exists more than T_o days and does not change to an m-cluster, we remove it from memory.
- Inverse covariance matrix initialization parameter γ. The parameter $\gamma > 0$ is used to initialize the inverse covariance matrix in a new cluster. Simulation results show that $\gamma = 10^8$ works well.
- Time parameter m. The time parameter m depends on how long it needs to update the cluster weights. For example, $m = 3$ means that the last 3 minutes of each day are used to update the cluster weights. New reports are not processed during that time.

More comments on parameter selection are made next in the simulation section.

E. Computation and Memory Efficiency

Computation and memory efficiency is one of the requirements specified in the Introduction section. Towards this end, the proposed clustering algorithm is designed that it processes anomaly reports in a single-pass fashion, that is, all reports are processed only once. All needed cluster information is maintained in the tuple specified in (4). The Woodbury Matrix Inversion Lemma is exploited to further simplify the cluster updates.

Consider now the computation and memory requirements in the scale of a city. Suppose N clusters exist in the city. For each cluster, all information needed to be stored is the five features specified in (4), which can be described using 9 single-precision floating point numbers (note that cluster center is a vector of dimension 2 and the covariance matrix is 2×2). Based on IEEE 754-2008 standard [20], each single-precision floating point number occupies 4 bytes. As a result, the total memory needed is $N \times 9 \times 4$ bytes. Let N be a million, the memory requirement is only 36 MB, which is very light for the scale of a city.

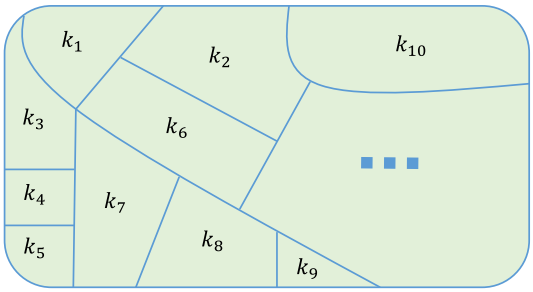


Fig. 4. Hierarchical strategy for cluster traversal.

The chronometric requirements are also light. Following Algorithm 1, when a new report arrives, we first follow Steps 7 and 8 in Algorithm 1 to traverse the N clusters and find the one closest to the new report. To accomplish this, we compute the Mahalanobis distance for all N clusters following Step 7 in Algorithm 1. After we find the closest cluster, we create a new cluster using Step 17 if Step 9 is not satisfied. Otherwise, we update features of the closest cluster using (9), (10), (13), (14) and Step 13 in Algorithm 1. These computations are all algebraic and do not involve matrix inversion (note that (14) is a scaler inversion). As a result, the complexity for processing a new report is $O(N)$.

The above computation process can be simplified by dividing the city into zones and instead of checking all the clusters, we first determine the zone where the report falls and we then only check the clusters within the zone. As illustrated in Figure 4, this hierarchical strategy can greatly reduce the number of clusters we need to traverse. If the city is divided into n zones and let k_i represent the number of clusters in Zone i, $i = 1, 2, \cdots, n$, then the worst-case complexity of the algorithm is further reduced to $O(n + K)$, where $K = \max_{i=1,\cdots,n} k_i$ represents the largest number of clusters in the zones.

IV. Simulation Demonstration

In this section, we present a simulation to demonstrate that our algorithm possesses the desired capabilities which we listed in the Introduction section. We simulate the algorithm on the roads around North Campus of the University of Michigan as illustrated in Figure 5. We include twelve anomalies to verify the ability of handling multiple clusters *without pre-defined number of clusters*. In order to show the ability of *localization of isolated anomalies and information compression for stretched anomaly segments*, the anomalies are arranged in a way that anomalies 1-5 and 8-10 are densely distributed while others are isolated . Four false alarms locations are added in the simulation to test the ability to *handle outliers*.

To demonstrate the ability of *handling anomaly evolution*, the algorithm is simulated over a period of 15 days with the following setup. From day 1 to day 5, anomalies 1-11 are present. On day 6, anomalies 8-10 are fixed, however, anomaly 12 appears. The total number of reports is uniformly distributed between 180 and 200 each day. The reports are randomly generated around the true anomalies with a covariance

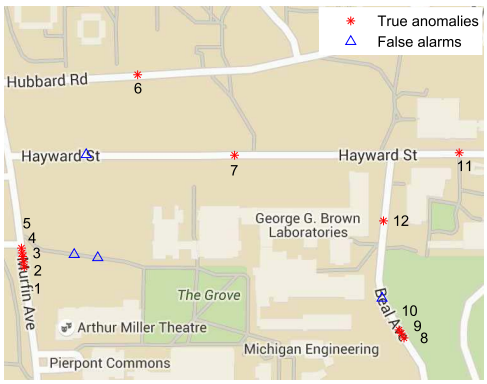


Fig. 5. Road anomalies and false alarms.

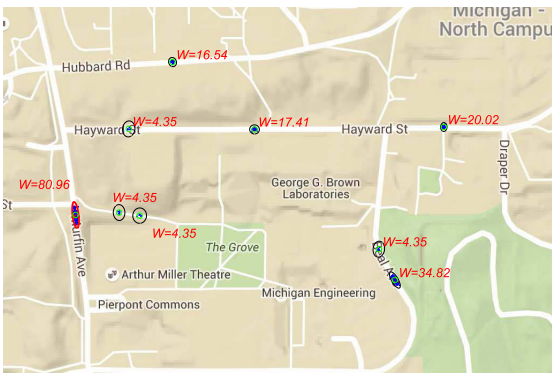


Fig. 6. Snapshot at the end of day 1.

TABLE II
PARAMETERS FOR SIMULATION

p	α	λ	h_o	h_m	T_o	γ
0.0027	2	0.2	80	50	5	10^8

matrix corresponding to a $5 \cdot \mathbf{I}_2\,(m^2)$ covariance in the state plane coordinates [19]. Five false alarm reports are generated around each of the four locations indicated as blue triangles in Figure 5, inducing an average false alarm rate in the anomaly data of 10.5%.

Note that parameters h_m, h_o, T_o, α, and λ depend on the road dynamics, e.g., traffic density, anomaly life cycle etc. and should be tuned cooperatively. If h_o is too high, we may discard true anomalies. At the same time, if h_o is too low, we may mishandle outliers and treat them as true anomalies. As for h_o, if it is too low, it takes a long time to discriminate fixed anomalies and is thus not memory-efficient. On the other hand, if it is too close to h_o, the o-clusters and m-clusters may interchange frequently. We thus require that $h_m < h_o \cdot \alpha^{-\lambda}$. The parameters T_o, α, and λ can then be similarly tuned to satisfy system requirements.

Parameters in Table II are used in the Matlab simulation. The p-value is chosen as suggested in Section III-D that corresponds to 3σ bands. In order to show that the outliers are discriminated over the 15-day period, we choose the pruning period $T_o = 5$. Based on the simulation setup, there are around 18 reports for each of the anomalies every day and h_o is set to 80 to allow the true anomalies to grow into m-clusters in $T_o + 1$ days (since we check the weight at the end of day). Note that if $h_o > 100$, then true anomalies will be discarded. The parameter α of the decaying function is set to 2 as suggested and the parameter λ is tuned so that the cluster formed by anomalies 8-10 changes to an outlier at the end of this simulation since they are fixed on Day 6. The parameter h_m needs to satisfy $h_m < h_o \cdot \alpha^{-\lambda}$ and it is selected as 50.

Note that anomalies 1-5 and anomalies 8-10 are groups that can be included in one cluster each for the benefits of memory efficiency. A snapshot at the end of day 1 is illustrated in Figure 6. With aggregated reports, the weight of each cluster increases. The cluster that covers anomalies 1-5 has a weight

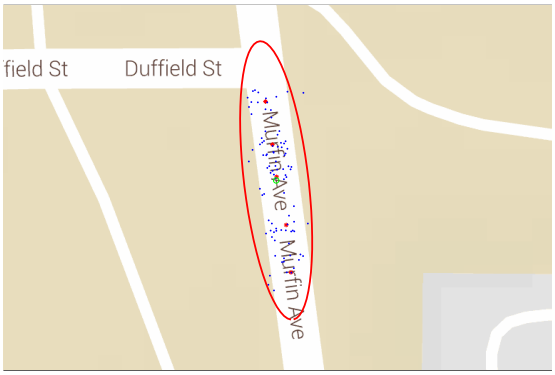


Fig. 7. Information compression for anomalies 1-5.

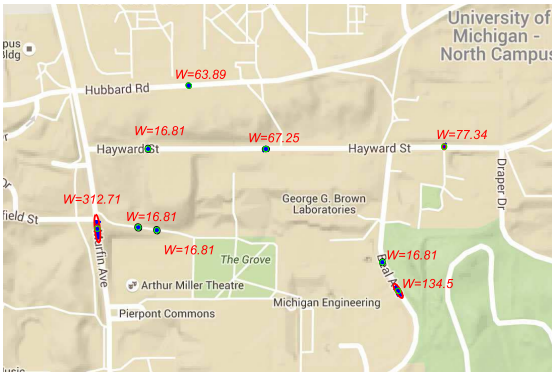


Fig. 8. Snapshot at the end of day 5.

of more than $h_o = 80$ and is labeled as an m-cluster with red ellipsoid. All other clusters' weights are below h_o and these clusters are labeled as o-clusters with black ellipsoid.

The clustering algorithm can compress information about densely distributed anomalies as illustrated in Figure 7. Anomalies 1-5 are included in one cluster that indicates a long stretch of road anomalies. This long stretch can be visualized on a map to warn drivers. Note that the maximum band can be compressed as it is controlled by the chi-square number $\chi^2_2(p)$. With a larger p, the algorithm is able to discriminate "closer" anomalies.

A snapshot at the end of day 5 is illustrated in Figure 8. With aggregated reports, the clusters formed by anomalies 1-5

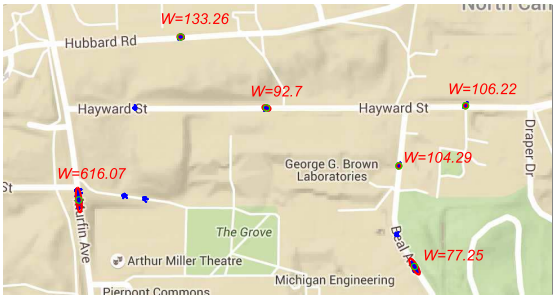


Fig. 9. Snapshot at the end of day 10.

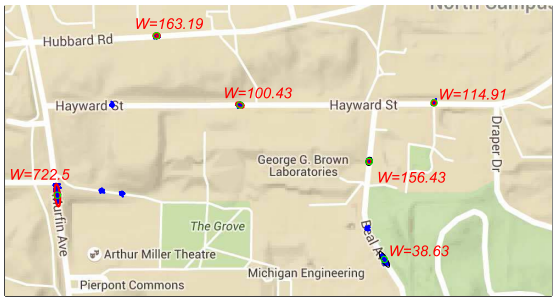


Fig. 10. Snapshot at the end of day 15.

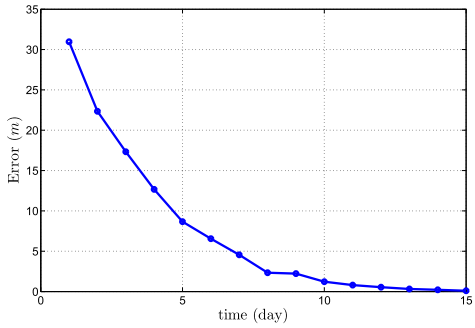


Fig. 11. Horizontal distance error of anomaly 6.

and anomalies 8-10 have greater weights than h_o thus they are m-clusters represented by red ellipsoids.

Based on the simulation setup from day 6, anomalies 8-10 are repaired and anomaly 12 is developed. A snapshot at the end of day 10 is shown in Figure 9. Since the outliers fail to become m-clusters in $T = 5$ days, the outliers are removed from the outlier buffer. This illustrates the algorithm's ability to deal with outliers. Also, since anomalies 8-10 are fixed and no new reports are aggregated, the weight decreases due to the decaying function.

Finally, a snapshot at the end of day 15 is shown in Figure 10. The weight of the cluster formed by anomalies 8-10 continues decreasing based on the decaying function. As a result, at the end of day 15, the weight becomes less than $h_m = 50$. The cluster is relabeled as an o-cluster and is moved to the buffer that is not shared with interested parties.

The horizontal distance error between the true anomaly position and the cluster center for anomaly 6 is illustrated in Figure 11. Note that with aggregated reports, the cluster center converges to the true anomaly location. Similar convergence results are obtained for the other isolated clusters.

The simulations show that the algorithm is able to handle outliers and can successfully capture the evolution of anomalies. As discussed in Section III-E, the computation and storage requirements are also light.

V. Conclusions and Future Work

In this paper, an anomaly report stream clustering algorithm was developed to process road anomaly reports that can be integrated into a Vehicle-to-Cloud-to-Vehicle framework. The cluster information was summarized in a feature vector that was recursively updated with new reports. The Mahalanobis distance was exploited to measure the similarity between a reported location and existing clusters. The obtained clusters are hyperellipsoids with arbitrary orientations. The Woodbury matrix inversion lemma was employed to facilitate the recursive computation of the covariance matrix inverse. The developed algorithm can reject outliers and capture the evolving road anomalies with a combined o-cluster and m-cluster strategy. The proposed algorithm can also localize isolated anomalies and compress information for stretched anomaly segments with light memory and computation requirements.

Future work will include a more comprehensive study of parameter selection based on real-world data.

References

[1] R. Mao and G. Mao, "Road traffic density estimation in vehicular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4653–4658.

[2] C.-S. Liu and H. Peng, "Road friction coefficient estimation for vehicle path prediction," *Vehicle Syst. Dyn.*, vol. 25, no. S1, pp. 413–425, 1996.

[3] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2008, pp. 29–39.

[4] Z. Li, I. Kolmanovsky, E. Atkins, J. Lu, D. Filev, and J. Michelini, "Cloud aided semi-active suspension control," in *Proc. IEEE Symp. Comput. Intell. Vehicles Transp. Syst.*, Dec. 2014, pp. 76–83.

[5] X. Yin, L. Zhang, Y. Zhu, C. Wang, and Z. Li, "Robust control of networked systems with variable communication capabilities and application to a semi-active suspension system," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 4, pp. 2097–2107, Aug. 2016.

[6] Z. Li, U. V. Kalabić, I. V. Kolmanovsky, E. M. Atkins, J. Lu, and D. P. Filev, "Simultaneous road profile estimation and anomaly detection with an input observer and a jump diffusion process estimator," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 1693–1698.

[7] Z. Li, I. V. Kolmanovsky, E. M. Atkins, J. Lu, D. P. Filev, and Y. Bai, "Road disturbance estimation and cloud-aided comfort-based route planning," *IEEE Trans. Cybern.*, [Online]. Available: http://ieeexplore.ieee.org/document/7514928/.

[8] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[9] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, vol. 29. 2003, pp. 81–92.

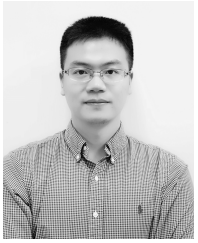
[10] T. N. Pappas and N. S. Jayant, "An adaptive clustering algorithm for image segmentation," in *Proc. 2nd Int. Conf. Comput. Vis.*, Dec. 1988, pp. 310–315.

[11] J. Grabmeier and A. Rudolph, "Techniques of cluster algorithms in data mining," *Data Mining Knowl. Discovery*, vol. 6, no. 4, pp. 303–360, Oct. 2002.

[12] D. Greene, A. Tsymbal, N. Bolshakova, and P. Cunningham, "Ensemble clustering in medical diagnostics," in *Proc. 17th IEEE Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2004, pp. 576–581.

[13] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer, 1981.

[14] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proc. IEEE Conf. Decision Control Including 17th Symp. Adapt. Process.*, Jan. 1979, pp. 761–766.
[15] N. Ailon, R. Jaiswal, and C. Monteleoni, "Streaming k-means approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 10–18.
[16] D. Filev and O. Georgieva, *An Extended Version of the Gustafson–Kessel Algorithm for Evolving Data Stream Clustering*. Hoboken, NJ, USA: Wiley, 2010, pp. 273–299.
[17] M. A. Woodbury, *Inverting Modified Matrices* (Statistical Research Group Memorandum Reports), vol. 42. Princeton, NJ, USA: Princeton Univ. Press, 1950.
[18] P. C. Mahalanobis, "On the generalised distance in statistics," *Proc. Nat. Inst. Sci. India*, vol. 2, no. 1, pp. 49–55, Apr. 1936.
[19] C. C. J. M. Tiberius and K. Borre, "Are GPS data normally distributed," in *Geodesy Beyond: The Challenges of the First Decade IAG General Assembly Birmingham*. Berlin, Germany: Springer, 2000, pp. 243–248.
[20] *IEEE Standard for Floating-Point Arithmetic*, IEEE Standard 754-2008, Aug. 2008, pp. 1–70.



Zhaojian Li (M'16) received the B.S. degree in civil aviation from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010, and the M.S. and Ph.D. degrees from the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA, in 2014 and 2016, respectively.

From 2010 to 2012, he was an Air Traffic Controller with the Shanghai Area Control Center, Shanghai, China. From 2014 to 2015, he was an Intern with Ford Motor Company, Dearborn, MI, USA. Since 2013, he has been a Graduate Research Assistant with the Department of Aerospace Engineering, University of Michigan. He is currently with General Motors Advanced Powertrain, Milford, MI. His research interests include optimal control, system modeling, estimation, and intelligent transportation systems. He was a recipient of the National Scholarship from China.

Dimitar P. Filev (F'08) received the Ph.D. degree in electrical engineering from Czech Technical University in Prague, Prague, Czech Republic, in 1979. He is currently an Executive Technical Leader in intelligent controls with Ford Research and Innovation Center, Dearborn, MI, USA. He is conducting research in modeling and control of complex systems, intelligent control, fuzzy and neural systems, and their applications to automotive engineering.

He holds numerous U.S. and foreign patents. He has authored four books and over 200 articles in refereed journals and conference proceedings. He was a recipient of the 2015 IEEE Computational Intelligence Society Pioneer's Award, the 2008 Norbert Wiener Award of the IEEE System, Man, and Cybernetics (SMC) Society, and the 2007 International Fuzzy Systems Association (IFSA) Outstanding Industrial Applications Award. He is the president-elect of the IEEE SMC Society. He is a fellow of the IFSA.

Ilya Kolmanovsky (F'08) received the M.S. and Ph.D. degrees in aerospace engineering, and the M.A. degree in mathematics from University of Michigan, Ann Arbor, MI, USA, in 1993, 1995, and 1995, respectively.

He is a Full Professor with the Department of Aerospace Engineering, University of Michigan. His research interests include control theory for systems with state and control constraints, and control applications to aerospace and automotive systems.

Dr. Kolmanovsky has been with Ford Research and Advanced Engineering, Dearborn, MI, for close to 15 years. He is named as an Inventor on 92 U.S. patents. He was a recipient of the Donald P. Eckman Award of American Automatic Control Council.

Ella Atkins (SM'14) received advanced degrees in aeronautics and astronautics from Massachusetts Institute of Technology, Cambridge, MA, USA, and in computer science and engineering from University of Michigan, Ann Arbor, MI, USA.

She is a Professor with the Department of Aerospace Engineering, University of Michigan, where she is also the Director of the Autonomous Aerospace Systems Laboratory. She has authored over 150 refereed journal and conference publications. Her research interests include task and motion planning, guidance, and control strategies to support increasingly autonomous cyber–physical aerospace systems.

Dr. Atkins was a member of the Institute for Defense Analyses Defense Science Studies Group from 2012 to 2013. She is an Associate Editor of *AIAA Journal of Aerospace Information Systems*. She also is on the National Academy's Aeronautics and Space Engineering Board. She is the Graduate Program Chair for the new University of Michigan Robotics Program.

Jianbo Lu (SM'09) received the Ph.D. degree in aeronautics and astronautics from Purdue University, West Lafayette, IN, USA, in 1997. He is currently a Technical Expert in advanced vehicle controls with Controls Research and Advanced Engineering, Research and Innovation Center, Ford Motor Company, Dearborn, MI, USA. He is an Inventor or Co-Inventor of over 100 U.S. patents. His invented technologies have been widely implemented in tens of millions of vehicles with brand names, such as Ford, Lincoln, Volvo, and Land Rover. He has authored over 70 refereed research articles. His research interests include automotive controls, intelligent and adaptive vehicle systems, integrated sensing systems, driving assistance and active safety systems, and future mobility.

Dr. Lu was a two-time recipient of the Henry Ford Technology Award at Ford Motor Company. He is an Associate Editor for IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He is on the Editorial Board of *International Journal of Vehicle Autonomous Systems* and *International Journal of Vehicle Performance*. He also is the Chair of the Intelligent Vehicular Systems and Control Technical Committee under the IEEE Society of Systems, Man, and Cybernetics. From 2008 to 2014, he was an Associate Editor of *IFAC Journal of Control Engineering Practice*. He is the Vice Chair for Industry and Applications at the 2015's American Control Conference.