Real-Time Traffic Incident Detection Using Probe-Car Data on the Tokyo Metropolitan Expressway

Akira Kinoshita
The University of Tokyo
Tokyo, Japan
kinoshita@nii.ac.jp

Atsuhiro Takasu and Jun Adachi
National Institute of Informatics
Tokyo, Japan
{*takasu, adachi*}*@nii.ac.jp*

***Abstract*—We have developed a real-time traffic incident detection system for the Tokyo Metropolitan Expressway. This system monitors current traffic using probe-car data and compares actual traffic in real time with the usual traffic, which is estimated in advance using batch processing.**

***Keywords*-anomaly detection; automatic incident detection; probe-car data; real data; real-time system;**

I. Introduction

Real-time automatic incident detection (AID) is crucial in solving traffic incidents early and reducing congestion. Probe-car data (PCD) are becoming increasingly important as the number of probe cars and the size of the data archives increase. With real-time PCD processing, early AID can be achieved for a vast area at a low cost.

In this paper, we develop a real-time traffic incident detection system that covers the whole of the Tokyo Metropolitan Expressway (MEX). The total length of MEX is about 300 km and the average daily traffic is about one million vehicles. Although the MEX forms the arteries of the Tokyo area, there are many bottlenecks, such as curves, and the speed limit is 60 km/h for the greater part of the routes. Traffic congestion is quite common there, and is not always caused by traffic incidents. In our previous paper [1], we proposed a method to detect incidents that we would regard as sudden and unusual traffic events by comparing between usual and current traffic states. This paper describes the design and implementation of the proposed system, which can apply our algorithm to the PCD stream in real time.

II. Detection Methodology

This section describes our incident-detection method, which we proposed in our previous paper [1] and modified to work in a real-time application. Our approach is to compare the usual and current traffic states to find traffic incidents, i.e., sudden and unusual traffic events. We first give a brief description of the traffic state model from our previous paper [1] that describes traffic states for a variety of roads. Then we describe our incident detection method based on the model, which issues an alert when a car's behavior is estimated to be sufficiently different from usual.

A. Traffic State Model

Intuitively, we can identify traffic states as "smooth" or "congested" regardless of location. Vehicles travel fast in smooth states and behave in a stop-and-go fashion in heavily congested states. When observing the speed of a probe car, the value is likely to be small if the car is in "congested traffic," or large if the traffic is "smooth." In short, the behavior of a car is affected by the surrounding traffic state and the observed values for the probe car will change, whereas the traffic state is latent and varies according to the time and place. We use a probabilistic topic model [2] to model this relation between traffic states and the PCD.

A traffic state can be associated with a probability distribution, which generates an observation value, e.g., speed. On the other hand, traffic states are strongly related to roads, so we introduce the *segment* as the unit for observing traffic. A segment is defined as a certain section on a route during a certain time period. Let K be the number of states, with the kth traffic state corresponding to the parameter $\boldsymbol{\theta}_k$. The probability distribution for the sth segment is described in terms of a mixture of these K distributions. The state parameters $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K\}$ are identical for all segments, but the mixing coefficient is different for each segment. The maximum-likelihood parameters of the model are estimated by an expectation-maximization (EM) algorithm, using archived PCD as training data.

B. Incident Detection

Our AID method measures the degree of anomaly for each probe car's trajectory. Figure 1 shows our concept to

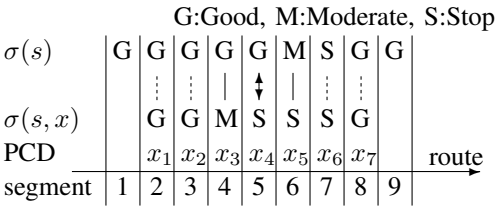

Figure 1. Concept of our incident detection method, which compares the usual and current traffic states.

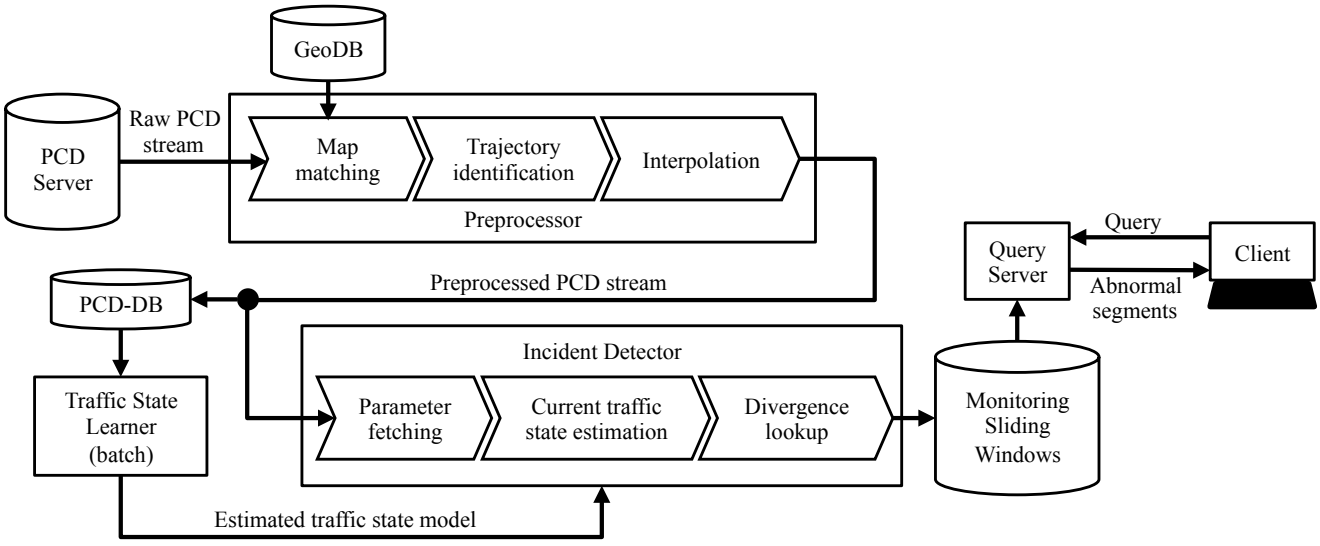

Figure 2. Architecture of our real-time traffic incident detection system.

detect incidents by comparing the usual traffic states with the current ones. Assume a probe car travels along a route and observes values for each segment that it passes through.

We first define the *current traffic state* when the value x was observed in the sth segment, denoted by $\sigma(s, x)$, as the most probable state given x. Using the posterior distribution with Bayes' theorem, $\sigma(s, x)$ is estimated as:

$$\sigma(s, x) = \arg\max_k \{\pi_{sk} p(x|\boldsymbol{\theta}_k)\}, \tag{1}$$

where π_{sk} is the mixing coefficient of the kth state in the sth segment. Meanwhile, the learned model itself reflects the usual state over the whole observation period, because the parameters are estimated to fit the distribution in the dataset. We can therefore define the *usual traffic state* for the sth segment, denoted by $\sigma(s)$, as the most probable state:

$$\sigma(s) = \arg\max_k \pi_{sk}. \tag{2}$$

We now introduce the *divergence* of $\sigma(s, x)$ from $\sigma(s)$, denoted by $d(s, x)$, to quantify the difference between the two states. Because in our model each state is associated with a probability distribution, we measure this difference in terms of the Kullback–Leibler divergence of the current state's distribution from the usual state's distribution. Finally, we define the divergence of a probe car's trajectory, denoted by D, as the summation of all the divergences $d(s, x)$ that were calculated using the car's observation values. The more a car deviates from its usual behavior, the larger D will be. A probe car's trajectory is determined as anomalous when D is sufficiently large, i.e., larger than a predefined threshold.

The D defined above will continue to increase as long as the car runs, and any car would eventually be determined as being anomalous. Therefore, in this paper, we redefine D as the sum of the N most recent divergences $d(s, x)$. If the car passes through fewer than N segments, the redefined D is equivalent to the original. This change allows the divergences of a probe car to be managed by a sliding window, i.e., a queue.

III. System Architecture and Implementation

Figure 2 shows the architecture of the proposed system, which applies our detection algorithm to the PCD in real time. Our system first preprocesses the raw PCD stream in three phases: map matching, trajectory identification, and interpolation. Map matching is a process to determine the segment that the probe car was in from the reported time and position. To reduce latency, our system conducts map matching in the simplest way: a probe car's observation is matched with the segment nearest to the car's location. We used PostGIS [3] to process this nearest-neighbor query. After map matching, the trajectory corresponding to the probe car's observation is identified, and linear interpolation is used to form an observation sequence of the consecutive segments through which the probe car passes. For this procedure, the preprocessor uses a hash table to store the last observation value for each trajectory. The output of the preprocessor is a stream of 4-tuples (trajectory ID, segment ID, time, speed), which are stored in a database.

Our detection algorithm is applied to the preprocessed PCD. The "Traffic State Learner" in the left part of Figure 2 is a batch processor that applies an EM algorithm to the stored PCD periodically, to estimate the parameters of our traffic state model. We implemented it using OpenMP [4] for multiprocessing. The "Incident Detector" applies our incident detection algorithm and it is implemented so that the data are processed in real time. It first obtains the estimated

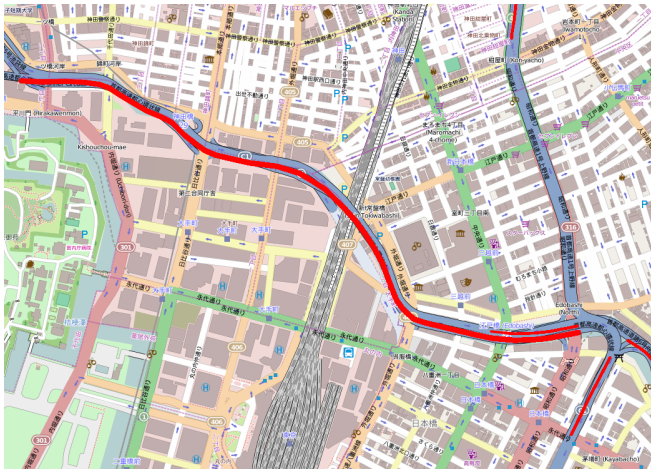

Figure 3. Screenshot of the monitoring screen showing the result of our real-time incident detection system. The red lines on the OpenStreetMap [5] indicate the segments with high divergence.

traffic state model and holds a hash table to find model parameters for each segment, and a lookup table to find divergences between any pair of traffic states. The detector also receives the preprocessed PCD stream. When a PCD instance arrives, the detector first fetches model parameters from the hash table and estimates the current traffic state. Then it fetches the divergence between the usual and current traffic states from the lookup table. Finally, the detector sends the divergence to "Monitoring Sliding Windows," which uses a hash table to access sliding windows, each of which holds recent divergence values of a trajectory. When a client issues a query to obtain abnormal segments, the "Query Server" reads the "Monitoring Sliding Windows" data and provides any detected segments.

We implemented a monitoring screen using OpenLayers 3 [6] to display the high-divergence segments on a map, as shown in the screenshot in Figure 3. The red line in the middle of the figure indicates that the estimated divergence of the segments between the Edobashi Junction and the Takebashi Junction on the inner loop (counterclockwise) of the Inner Circular Route was high. Traffic congestion occurred at that time and a probe car ran at unusually slow speed, so our system estimated the divergence to be high. According to the traffic log made available by the administrator of the MEX, this congestion was actually caused by a traffic accident.

IV. Discussion

Our prototype system runs with static parameters at present. The parameters include the lengths of road segments, the number of traffic states K, and the length of the sliding window N. However, they should be updated adaptively for a real-time application because the optimal parameter values might vary according to the traffic. We will introduce parameter-tuning functions to improve the detection accuracy in the next phase.

We also plan to apply our system to a larger road network. Reducing memory usage of the system is considered vital to achieve scalability. The "Incident Detector" uses memory to hold a hash table and a divergence lookup table. The size of the hash table is proportional to the number of segments and K, and the size of the lookup table is $K \times K$. As we found in our previous paper [1], K is a rather small number, e.g., eight. Therefore, the memory usage is approximately proportional to the number of segments. The current system partitions the MEX routes to constant-length segments, but the number of segments can be reduced by varying their lengths according to the location and traffic while preserving the detection accuracy. Improving throughput for each process is also vital, and the performance of the system is under study.

V. Conclusion

We have developed a real-time traffic incident detection system that covers the whole of the Tokyo Metropolitan Expressway. This system processes probe-car data streams to apply our detection algorithm [1] in real time. Future work is under way to update parameters adaptively and to reduce memory usage of the system so that the system will work more efficiently in real time and for larger road networks.

Acknowledgment

This work was supported by CPS-IIP Project in the research promotion program for national-level challenges "Research and development for the realization of next-generation IT platforms" by the Ministry of Education, Culture, Sports, Science and Technology, Japan. The traffic log used in our experiment as the ground truth for incident occurrence was made available by Metropolitan Expressway Co., Ltd.

References

[1] A. Kinoshita, A. Takasu, and J. Adachi, "Traffic incident detection using probabilistic topic model," in *Proc. Work. EDBT/ICDT 2014 Jt. Conf.*, 2014, pp. 323–330.

[2] D. M. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012.

[3] PostGIS, http://postgis.net/.

[4] OpenMP, http://openmp.org/wp/.

[5] OpenStreetMap, http://www.openstreetmap.org/.

[6] OpenLayers 3, http://openlayers.org/.