How Many Bumps in Your City? Personalized Bump Seeker With Mobile Crowdsensing

Xuan Xiao, Ruipeng Gao, Weiwei Xing, *Member, IEEE*, Chi Li, and Lei Liu

***Abstract*— Decorating speed bumps is a common and effective measure to compulsively reduce vehicular velocity in dangerous regions, especially at tunnels, slopes, and parking lots. Despite a decade of deployment, the bump information is still sporadic in map-based applications, for example, driving alert ahead of bumps. One major obstacle is the lack of an up-to-date bump database, thus service providers have to hire dedicated personnel to gather and periodically calibrate such data, which is effort-intensive and time-consuming to large-scale coverage. In this article, we propose a personal bump seeker (PBS), a novel mobile crowdsensing application to automatically identify and update speed bumps in urban cities. Specifically, we formulate bump detection as a regression model, thus improving the robustness of inertial patterns on temporal domain. We also explore a leader-follower mechanism that automatically extracts the bump signal for training and inference in individual smartphones, regardless of different smartphones, vehicles, and driving habits during crowdsensing. Finally, we implement a prototype and conduct extensive experiments on large-scale real-world traffic datasets collected by the DiDi platform, and the results demonstrate our effectiveness in producing a city-level bump database.**

***Index Terms*— Crowdsourced measurement, leader-follower mechanism, personalized training, speed bump detection.**

I. Introduction

DEPLOYING speed bumps [see Fig. 1(a)] has been an effective administrative measure for driving safety over a period of decades. Drivers have to slow down vehicles to get rid of the sharp turbulence when passing by, thus focusing their attention on accident-prone areas such as steep slopes, dim tunnels, parking lots, and so on.

However, such bump information is extremely scarce and unavailable in most maps. This has become a huge obstacle to intelligent transportation systems.

1) For transportation administration, the bump database satisfies the fundamental cognitive needs to understand road conditions. For example, we care about whether the traffic congestion is caused by excessive installations of speed bumps, or if a new bump should be placed in an intersection where traffic accidents often accrue.
2) For individual drivers, the bump information not only supports early driving alert, but also calibrates the vehicle's location in global positioning system (GPS)-blocked environments (e.g., vehicle tracking [1], [2] in underground parking structures), thus improving our driving experience intensively.

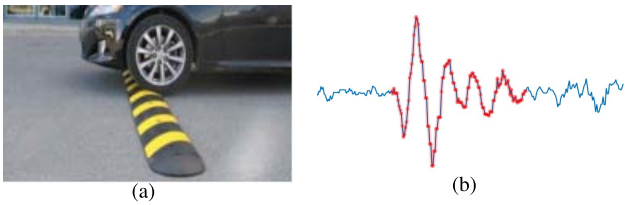


Fig. 1. (a) Speed bump. (b) Accelerations along gravity direction when passing the bump.

Constructing a complete and up-to-date bump database at low cost is urgently needed. Before the popularity of smartphones, [3], [4] utilized special sensors to measure the road condition. High-precision special sensors can identify and measure the road condition with high quality, at the cost of manufacture which makes it difficult for large-scale deployment. Recent works [1], [5]–[8] have leveraged the inertial data from smartphones to detect speed bumps, that is, via the fluctuated acceleration along gravity direction [see Fig. 1(b)]. However, most of them utilized dedicated datasets instead of real-world crowdsourced data, thus escaping from combating inevitable errors and noises in crowdsensing. Other approaches use images [9]–[11] but also face common limitations in vision: dark/changed lighting, blurry images, glass walls, and restrictions on photo-processing due to computation ability.

In this article, we propose personal bump seeker (PBS), a novel personalizing-based system for city-level bump database construction using crowdsourced data from commodity smartphones. Unlike using a general model or fixed thresholds to recognize each bump, we explore a personalized model training mechanism which is capable of producing very adaptable bump observations. Unlike images, its performance is not affected by lighting conditions or weather, nor does it occupy too much computation resource. Bumps in city level can be identified in a few hours by crowdsourcing, eliminating huge cost for humans.

Despite its potentials, constructing resilient and robust bump database is far from straightforward. First, the duration of bump signals varies with different vehicles and driving

Manuscript received November 2, 2021; accepted November 29, 2021. Date of publication December 15, 2021; date of current version March 1, 2022. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2021YJS186 and Grant 2021QY011, in part by the National Natural Science Foundation of China under Grant 61876018 and Grant 62072029, in part by the Beijing NSF under Grant L192004, and in part by the DiDi Research Collaboration Plan. The Associate Editor coordinating the review process was Dr. Valentina Bianchi. *(Corresponding author: Weiwei Xing.)*

Xuan Xiao, Ruipeng Gao, and Weiwei Xing are with the School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: xiaoxuan@bjtu.edu.cn; rpgao@bjtu.edu.cn; wwxing@bjtu.edu.cn).

Chi Li and Lei Liu are with DiDi Company, Beijing 100085, China (e-mail: lichi@didiglobal.com; liuleifrey@didiglobal.com).

Digital Object Identifier 10.1109/TIM.2021.3135549

behaviors, thus it is difficult to set a uniform time window for binary classification. Second, in order to customize a model for each individual for accuracy, their data should be automatically annotated, instead of effort-intensive and time-consuming manual annotation by humans. Finally, both the inertial signal and the global position of bumps are required to update the map database for mobile applications.

Considering the above challenges, our contributions include as follows.

1) We propose a novel regression method for bump recognition based on inertial readings. It devises the bump duration probability to formulate the progress of a vehicle passing over the bump, thus enhancing the model adaptability.
2) We propose a leader-follower mechanism to improve the recognition accuracy through model customization. To produce the training data automatically without user's participation, we effectively align and extract the most similar signal segment with known patterns.
3) We propose a hidden Markov model (HMM) to detect bumps based on the sequence of bump duration probability. We further extract the bump signal and record its associated locations on the map.
4) We conduct experiments on two datasets: dedicated data with manually labels, and large-scale real-world data collected by a DiDi ride-hailing platform. Results have shown our effectiveness.

The architecture of our system will be depicted in III. In Section IV, we will describe the data processing flow and the recognition model we designed. The method of personalizing a general model is presented in Section V. Section VI shows the inference procedure of the placement of bumps. Section VII introduces our experimental design, results, and analysis of the results in detail.

II. Related Work

In the field of road condition detection, methods usually are divided into vision-based methods and inertial measurement unit (IMU)-based methods.

A. Vision-Based Methods

Vision-based methods usually use cameras or traffic monitoring cameras placed on vehicles as data collection devices to classify and locate road conditions through image or video processing. With the development of artificial intelligence, machine learning and deep learning methods are widely used in this field.

Schiopu *et al.* [9] proposed a threshold-based algorithm to extract candidate regions on frames and used a decision tree to classify the road condition. Fan *et al.* [10] used a supervised, convolutional neural network (CNN)-based method to recognize different road conditions on pavement textures. Arya *et al.* [12] used CNN methods to detect road damage. They also use transfer learning to customize detection models for other cities. Quintana *et al.* [11] collect the image of the road from a single camera on a light truck and use support-vector machine (SVM) to detect and classify cracks on road surfaces. However, the vision-based methods have limits on bad weather. Image processing is very susceptible to rain, snow or cloudy or light, and other environmental factors. Among video-based processing schemes, the influence of the device angle on the actual results is mentioned, for example, video processing needs to ensure that the device maintains a fixed angle. More than that, another obvious limitation of the visual method is the need for labeled image data. Image annotation is time-consuming and labor-intensive work.

B. IMU-Based Methods

The other method for road condition detection is IMU-based, that is, accelerometers and gyroscopes. When sensors deploy on driving cars, the bumps and potholes will reflect on IMU readers. Road condition methods based on IMU are mainly in two ways: detection based on a threshold and detection based on the machine learning method.

Pothole Patrol System (P2 approach) [3] is an earlier road surface monitoring method based on mobile sensors. They manually annotate the abnormal signals and use filters to control the thresholds for rejecting the "non-pothole" events. Mednis *et al.* [13] improved the P2 approach. They proposed more specific threshold-based algorithms: Z-THRESH (improvement approach of z-peak threshold detection in Pothole Patrol [3]) to threshold the acceleration amplitude at Z-axis, Z-DIFF to detect fast changes in vertical acceleration data, STDEV(Z) to calculate the standard deviation of vertical axis acceleration, and G-ZERO to judge whether all the three-axis values are below the specific threshold level. However, threshold-based methods are always ineffective because of the diversity of abnormal signals.

With the development of machine learning, some researchers use machine learning methods to detect road conditions. Mohamed *et al.* [14] proposed an intelligent road detection system. The smartphone is placed in the car to collect acceleration and GPS data, and the data is pre-processed using a second-order high-pass Butterworth filter. Cross-validation is applied for acceleration and GPS data. In the classification stage, an SVM classification algorithm and two different kernel functions are used to classify the road surface into two categories: smooth and bumpy. El-Wakeel *et al.* [6] collected the accelerometers and gyroscopes data from smartphones attached to the driving cars. Then they developed a wavelet packet de-noising method to enhance the quality of data. After the feature extraction from statistical, time domain, and frequency domain, they applied the SVM method on the data to classify the road condition. To get better classification results, [5] expounds the Random Forest method in road condition detection tasks. They extracted features from different frequency domains. The Random Forest method exhibited the best classification performance for potholes, with a precision of 88.5% and recall of 75%.

Some methods use dynamic time warping (DTW) to enhance classification accuracy. Singh *et al.* [7] improved the classification accuracy of detecting road surface conditions

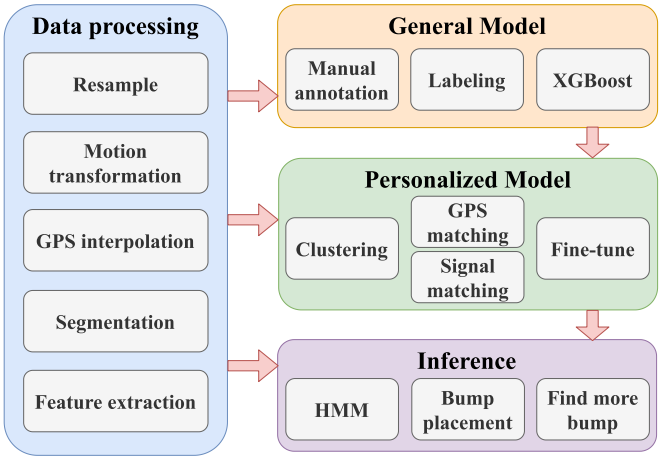


Fig. 2. System architecture.

by using the DTW2 technique. The main features of DTW are its ability to automatically cope with time deformations and different speeds associated with time data. The simplicity of DTW is adaptive to resource-constrained devices such as smartphones, and the training procedure is as fast as techniques such as SVM, HMM, and artificial neural network (ANN). Also, [8] presents the method using IMU of smartphone and crowdsourcing. They used DTW to compare two time-dependent series to enhance the accuracy and performance of road condition detection. After applying a brief random forest filter to distinguish anomaly data, [15] proposed a system based on DTW that utilizes a series of acceleration data to discover where might be anomalies on the road, named quick filter-based dynamic time warping (QFB-DTW).

However, it is difficult for general recognition patterns or recognition models to recognize large-scale crowdsourced data.

III. System Overview

A. Architecture

In this article, we propose PBS that effectively utilizes the inertial data from crowdsourced in-vehicle smartphones to seek road bumps at city level. Fig. 2 depicts our system architecture which is comprised of data processing, general model training, personalized model training, and personalized on-device inference.

First, we establish a bump database through a small amount of manually annotated data and train a general bump detection model. In order to identify bumps for individual drivers despite their specific vehicle types and driving behaviors, we train their personalized models based on their associated bump signals when they pass the annotated bumps (with GPS positions). Finally, these crowdsourced drivers go through the whole city, with their personalized model to seek new bumps and augment the database.

IV. General Model Training

A. Segmentation and Feature Extraction

For both training and inference, the raw inertial data will undergo segmentation and feature extraction operations before feeding into the model.

1) Sliding Window: To provide inertial sequences with fixed size for model training, we set the window size as 1.5 s and the step length as 0.4 s. The effect of different window sizes are analyzed in Section VII.

2) Features: For sequential data, features extracted from time domain, frequency domain, and wavelet domain have been proved to be effective [16]–[18].

1) Time-domain features reflect the vibration of the signals. In addition to common ones such as mean, median, variance, standard deviation, maximum value, and minimum value, we also calculate statistical variables such as the RMS, range, skewness, and kurtosis [19]. Thus, there are ten statistical variables in our time-domain features.
2) Frequency-domain features are better at expressing the variation amplitude of signals. We conduct fast Fourier transform (FFT) operation to derive the frequency information and calculate ten statistical variables as the frequency features.
3) Discrete wavelet transform (DWT) decomposes the signal into several subband wavelets. We follow [5] and choose the Reverse Biorthogonal 3.1 wavelet to extract a list of 30 (10 statistical variables $\times$ 3 subbands) features.

We have discussed the effect of different domain features in Section VII.

B. Model Design

1) Bump Duration Probability: We explore a metric of bump duration probability to describe the progress of a vehicle passing over the bump.

It is defined as the probability of a bump signal within the time window, that is, the overlap of the window and the bump signal. It can avoid to determine whether the edge of a bump signal is classified as a bump, but using the continuous probability instead. When the time window slides with the sequential signal, this issue belongs to a typical regression problem.

2) Label Design: Fig. 3 shows that as the time window slides, there is an increasing overlap area L_{overlap} between the time window and the bump signal (annotated manually). Note that since the duration of a signal (L_{signal}) varies significantly via crowdsensing, it may be larger or smaller than the fixed time window (L_{window}). Thus, we define the segment label as the fraction of overlap area within the smaller one of L_{signal} and L_{window}, that is,

$$\text{label} = \frac{L_{\text{overlap}}}{\min\left(L_{\text{window}}, L_{\text{signal}}\right)}. \tag{1}$$

This formulation ensures that the label distribution is within [0, 1], even the time window may not cover the whole signal.

3) Training: We employ the XGBoost [20], [21] algorithm to learn the duration probability from manually annotated bump segments. XGBoost is an optimized distributed gradient boosting library which is highly efficient, flexible, and portable. Its objective is set as regression with a gbtree booster in our system. In addition, XGBoost is able to conduct continually training based on existing models, which is suitable for personalized training on smartphones. For general training, we take 200 epochs to ensure the model well trained.

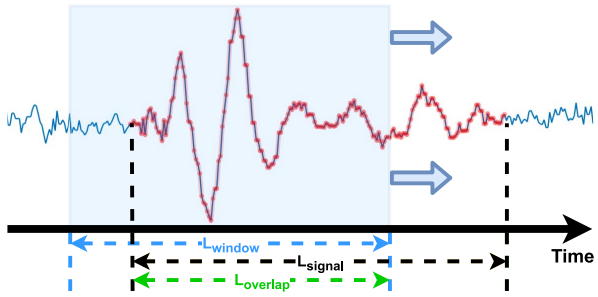


Fig. 3. Overlap between time window and bump signal.

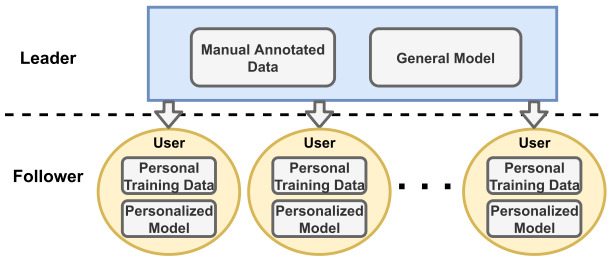


Fig. 4. Architecture of the leader-follower mechanism.

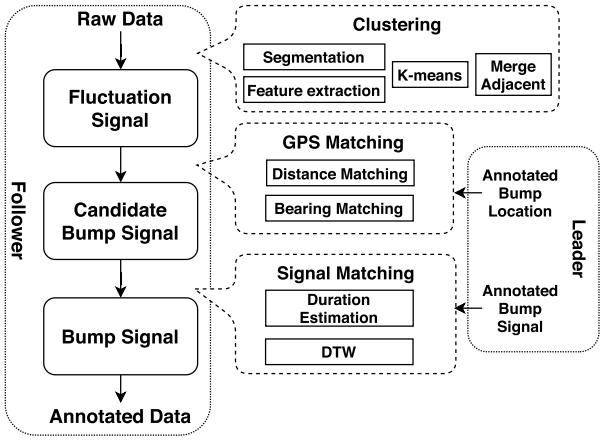


Fig. 5. Process of data auto-annotation.

V. Personalized Training

In order to identify bumps for crowdsourced drivers, we explore a leader-follower mechanism to train a personalized detection model for each individual. The architecture is shown in Fig. 4. The leader is equipped with a few manually annotated data and a generally trained model. When followers encounter annotated bumps by leaders, we extract their corresponding bump data (i.e., data auto-annotation) and train their own personalized model (i.e., model fine-tuning).

A. Data Auto-Annotation

As shown in Fig. 5, the data auto-annotation contains three steps to produce personalized data for training, including clustering, GPS matching, and signal matching.

1) Clustering: We design a clustering algorithm [22] to identify the fluctuation signal in the raw data. Specifically, we use a small time window of 1 s and a long step length of 0.9 s, and extract features including Mean, Range, and Std, which are different from the parameters in Section IV-A. Next, we cluster these data segments by the k-means algorithm ($k = 2$) to distinguish smooth and fluctuant data and merge adjacent fluctuate signals of neighboring time windows.

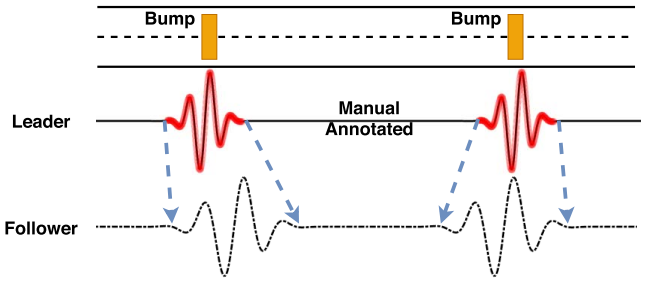


Fig. 6. Diagram of signal wave matching.

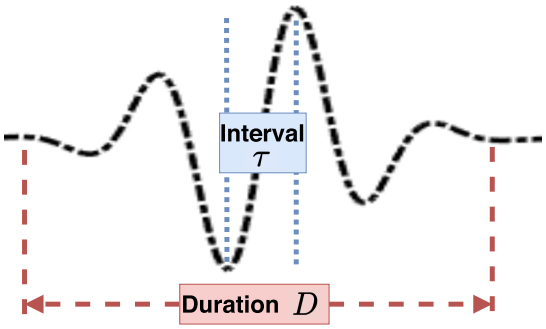


Fig. 7. Interval represents the length between two peaks and duration is the length of the whole bump signal.

2) GPS Matching: Since the GPS location of a few bumps is annotated by the leader, we use them to remove the interference (e.g., hand movement) in fluctuation signal segments. For every segment in fluctuation signal, if there is an annotated bump within a small distance (e.g., <10 m) and with the same orientation (<45°), we regard this segment as a candidate bump signal.

3) Signal Matching: Due to the diversity of vehicles and driving behaviors, a bump signal may fluctuate at different duration between leaders and followers (see Fig. 6). Thus, we explore a two-step algorithm to align their bump signals and remove outliers from the candidates.

Step 1: Bump duration estimation. We estimate the duration of each bump signal in candidate bump signal. In practice, we observe that there is a proportional relationship between the duration of bump signal and the time interval between two adjacent signal peaks (shown in Fig. 7). Thus, we calculate the follower's signal duration D_f as

$$D_f = \frac{\tau_f}{\tau_l} \times D_l \tag{2}$$

where τ denotes the time interval between two adjacent signal peaks, and D_l represents the signal duration from the leader.

Step 2: Signal alignment. We adopt a DTW [23], [24] algorithm to align the bump signal between leaders and followers. By the bump duration we estimated, we can segment a signal from the follower and take it as input of DTW together with the corresponding one in the leader. DTW stretches or

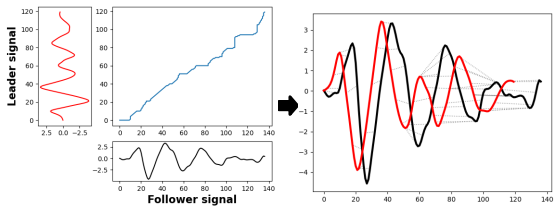


Fig. 8. Example of DTW. Left: Asymmetric alignment between two signals. Right: A pointwise comparison of two signals. The distance of points is calculated in Euclidean.

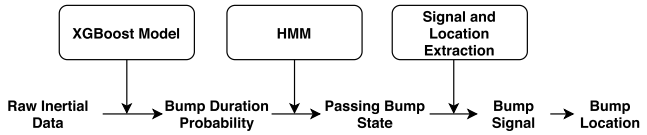


Fig. 9. Process of inference.

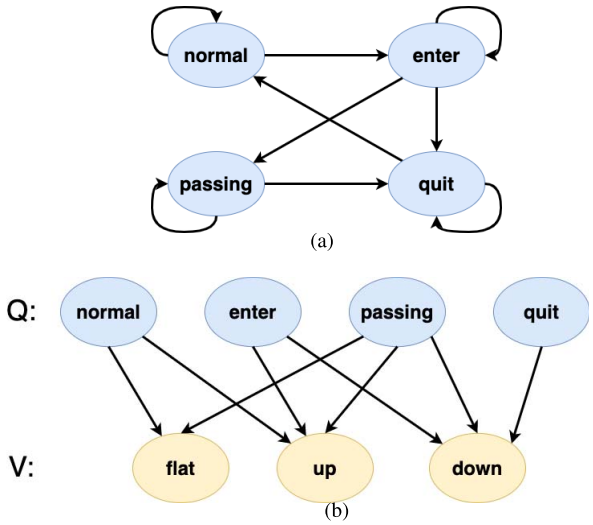


Fig. 10. (a) State transition relationship. (b) Output observation.

compresses two signals to make the signal of the follower resemble the leader one as much as possible and then aligns every element of them (shown in Fig. 8). After that, by summing the distances of every pair of aligned elements, we can get an indicator to measure the similarity of two signals: the smaller the distance summation, the more similar they are. Finally, we use a sliding window over the candidate signal to derive a signal segment with the highest similarity.

B. Model Fine-Tuning

In order to produce a personalized model, we exploit the personal training data to fine-tune the general model. The procedure is similar to the training of a general model. First, the annotated data is labeled by the method in Section IV-B. Second, we train a new XGBoost model based on the general model. Since the general model has already been trained, the training process of fine-tuning will not take much time, for example, 50 epochs are tested to be efficient.

VI. Inference

Based on the bump duration probability produced by general and personalized models, we further extract precise bump signals from inertial data and estimate their locations on the map. As Fig. 9 shows, the personalized model uses an XGBoost algorithm to generate a sequence of bump duration probabilities (see Section IV) and we formulate an HMM [25] to depict the process of passing a bump. Next, we identify and extract the complete bump signal and record its location based on crowdsourced GPS trajectories.

A. Hidden Markov Model

We exploit the bump duration probability to formulate the process of passing a bump. Setting a threshold as is can help us to identify the bump, but the threshold varies for different smartphones and vehicles. Based on the temporal correlations of duration probabilities over a sequence, we propose an HMM to inference personalized bump behaviors.

1) Hidden States: When a vehicle passes over a bump, it goes through four movement states, for example, driving on the flat road, encountering the bump, passing the bump, and leaving the bump. These vehicle states are defined as the hidden states in HMM, let Q be the set of all possible states, that is,

$$Q : \{\text{enter, quit, passing, normal}\}. \tag{3}$$

In addition, our aim is to precisely identify the passing state, which indicates the vehicle is locating on the bump. The transition relationship of each state is shown in Fig. 10(a). Every state has a probability to repeat itself, and the regular transition order of passing a bump is normal, enter, passing, quit, and normal. Note that we allow the state transition from enter to quit, which plays an important role in filtering the small vibration.

2) Observation States: Since we use the sliding window to segment inertial data, the variation along the duration probability sequence implicitly indicates the process of passing a bump. For example, Fig. 11 shows that the duration probability varies when the time window slides, with the observation states of Flat, Up, Flat, Down, Flat on its variation, and our aim is to estimate the corresponding hidden states. Thus, there are three observation states in HMM, that is,

$$V : \{\text{up, down, flat}\} \tag{4}$$

where up means the bump duration probability increases, down is the opposite, and flat means the variation of duration probability is rare. The relationship between hidden states and observation states are shown in Fig. 10(b). A flat state can be caused by normal and passing states, while up usually caused by enter and quit for down.

3) Problem Formulation: A typical HMM involves the hidden states Q, the observation symbols V, and specification of three probability densities A, B, and π. A is the state transitional probability matrix [see Fig. 10(a)], B is the observation probability matrix [see Fig. 10(b)], and π is the initial state probability. After obtaining the observation sequence

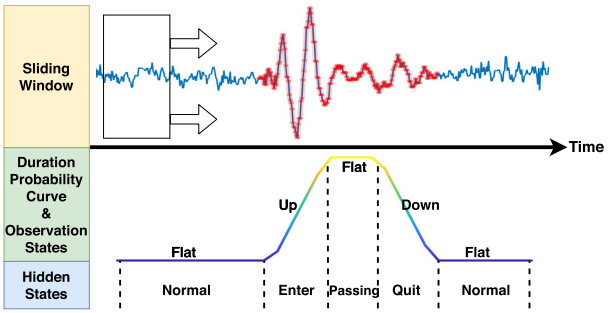


Fig. 11. Example of an HMM. Top layer shows that a slide window goes through the raw signal. The curve in the second layer is the corresponding output with the observation states. The last layer shows the corresponding hidden states.

$\mathbf{O} = (o_1, o_2, \ldots, o_T)$ from the bump duration probability sequence, we estimate the state sequence $\mathbf{I} = (i_1, i_2, \ldots, i_T)$ to achieve the highest probability, that is,

$$\mathbf{I} = \arg\max_{1 \leqslant t \leqslant T} P(\mathbf{O}|\mathbf{I}), \quad o_t \in V, \quad i_t \in Q. \tag{5}$$

Finally, we adopt the Viterbi algorithm [25] to derive the best state sequence.

B. Bump Placement

After detecting the bump-passing states for each driver, we further extract its bump duration and estimate its location on the map.

1) Signal Extraction: According to the bump duration probability, we calculate the start and end time of the bump signal. If there are continuous windows that go through a bump signal, we use the first window to calculate the start time of the bump signal, and the last one calculates the end time. Suppose that the start time of the first window is t, the length of this window is l, and the bump duration probability is p, then we calculate the start time of the bump signal segment by $t + l \cdot (1 - p)$. The end time calculates similarly, with the formula: $t + l \cdot p$.

2) Placement: Since we collect both inertial data and GPS locations as inputs, we segment the bump signal over its duration and place its sequence on the map. Instead of computing its center point as the bump location, we choose the peak point of inertial data as its location, which is resilient and robust to inertial noises.

VII. Experiment and Evaluation

A. Datasets

Two kinds of datasets are adopted to support our experiment and evaluation: dedicated data and crowdsourced data. Differences exist in labeling status, data sources, usages, evaluation contents, and so on. Details for our datasets are shown in Table I.

1) Dedicated Data: These data are collected and labeled carefully. The collection steps are as follows.

1) We develop an application that can collect data from built-in hardware (accelerometer, gyroscope, GPS, etc.) in a smartphone. The data will be stored in a specific format in local storage in real-time.

TABLE I
Datasets Information

Dataset	**Dedicated**	**Crowdsourced**
Source	Our prototype	DiDi platform
Distance	168 km	2745 km
Time	5.7 hours	50.8 hours
Users	12 users	305 users
Sample rate	50 Hz	40 - 200 Hz
Ground truth	Signal segment	GPS position

2) In order to collect and label bump signals, we use two smartphones to gather the inertial signals and manually mark (click on the screen) the GPS position of bumps, respectively. The reason for using two smartphones is that the click event may involve extra inertial fluctuations. In addition, the timestamp synchronization between two smartphones is only acquired to associate inertial signals to bumps, which is acceptable within 1-s errors.
3) We collect data for drivers with different driving habits (some driving rough, some driving smooth). These data are stored independently in units of trajectory.
4) We manually annotate the beginning and ending timestamps of each bump signal in all data.

In virtue of the fine annotated, dedicated data can support the training of models and most experimental contents. However, the inevitable problem is that this part of data has a quantitative disadvantage. For this reason, we introduce crowdsourced data.

2) Crowdsourced Data: Our large-scale real-world crowdsourced dataset is collected by the DiDi platform. The amount of data is large while the quality is not high. The data is divided into trajectories, and each trace represents the track of a vehicle in a continuous period. However, it is hard to annotate crowdsourced data like dedicated data, and we even do not know the ground truth of crowdsourced whether they go through bumps. So, we design a method to count all the bumps a trajectory passes through, which we will describe in detail in Section VII-C2.

3) Information of Bumps: Based on bump material and road condition, we have classified six common types of speed bumps.

From a view of material, we identify three types of bumps: rubber bumps, metal bumps, and cement bumps.

1) Rubber bumps trigger slight noise and strong shock absorption when vehicles passing by and easily to be damaged. It is the most common speed bumps on low-speed roads.
2) Metal bumps provide long service life and satisfactory damping effect and are ordinary on all kinds of roads including highways.
3) Cement bumps are low cost, compression resistant, difficult to deform, but provide poor shock absorption.

From a view of road conditions, we also identify three types of bumps: on a straight road, at a road turning, and along a slope.

TABLE II
INFORMATION OF BUMPS

	Quantity	# Signals
Rubber	38	295
Metal	17	110
Cement	22	111
Straight	53	403
Turning	27	87
Slope	5	26

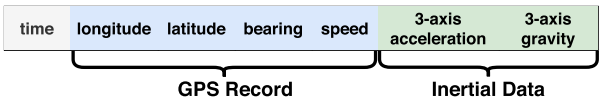


Fig. 12. Data format.

1) On a straight road: it is the most common road condition, where vehicle's two front wheels pass the bump at the same time, then comes two rear wheels.
2) At a road turning: we observe that vehicle's two front wheels roll over successively, but its two rear wheels generally pass at the same time.
3) Along a slope: it is not a general bump but consists of continuous embossing stamps to remind drivers to slow down. In addition, they exist more on downhill rather than uphill.

We add these definitions to the bump database and count the quantity of each type. The statistical result is shown in Table II.

In Table II, Quantity represents the number of each bump type, and # Signals represents the count of bump signals. Since vehicles may pass the same bump for multiple times, the values of # Signals are more than that of Quantity.

B. Data Processing

There are three parts for data segments during data collection, including the time stamp, GPS records, and inertial data. Detailed data format is shown in Fig. 12.

1) GPS Records: GPS records contain the vehicular location (longitude and latitude) and orientation (bearing): 1) since the sampling rate (1 Hz) of GPS is always lower than that of the inertial data (50 Hz), we perform 1-D linear interpolation operations on the GPS sequence to attach the location information on each frame of inertial data and 2) the bearing value is calculated via vehicle's movements, and it will be Null if the vehicle stops. Thus, we pad such missing values with its nearest available neighbor.

2) Inertial Data: We exploit the acceleration variation along gravity direction to identify the bump signal. However, there are several technical challenges when collecting such data in practice.

1) The sampling disturbances. Since it is difficult to maintain the same sampling frequency for all sensors, we resample and interpolate them at the fixed frequency of 50 Hz.

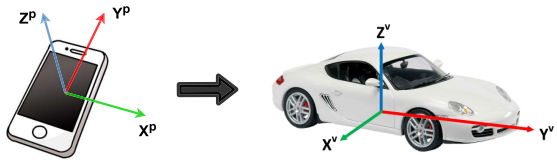


Fig. 13. Coordinate system transformation from the phone to the vehicle.

2) The phone's arbitrary placement in car. Although we could transform the 3-D movement from phone's coordinate system to that of the vehicle (see Fig. 13), we observe that the acceleration along gravity direction is the most relevant signal, thus we leverage the gravity API to extract the acceleration along vehicle's z-axis for efficiency.

C. Evaluation Metrics and Methods

To evaluate the performance of our methods, four indicators will be used: precision, recall, F1-score, and IoU [26]. As we compare the prediction results obtained from the method and the ground truth, precision can tell us whether the prediction result is correct, recall indicates that the proportion of bumps we miss found, and F1-score, which is a balance metrics of precision and recall can generally reflect the comprehensive performance of the method

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{F1-score} &= \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}. \end{aligned} \tag{6}$$

In addition, our method can find out the signal segment of bumps, we bring in IoU which can measure the accuracy of bump signals extraction.

1) Evaluation Metric of Signal Extraction: This evaluation method is suitable for dedicated data. For manually labeled data, we know the truly start (S_{gt}) and end (E_{gt}) time of the bump signal. When we extract the signal by the algorithm, that is, we get the start (S_{pred}) and end (E_{pred}) time of the bump signal, we can calculate the metrics. A predicted bump is regarded as TP, which means its ($S_{\text{pred}}, E_{\text{pred}}$) overlap with the corresponding ($S_{\text{gt}}, E_{\text{gt}}$). FP means that no overlap. If any bump in the ground truth fails to find the corresponding one in the prediction, the count of FN will increase. For all the bumps, we calculate the IoU to evaluate the accuracy of signal extraction

$$\text{IoU} = \frac{(S_{\text{gt}}, E_{\text{gt}}) \cap (S_{\text{pred}}, E_{\text{pred}})}{(S_{\text{gt}}, E_{\text{gt}}) \cup (S_{\text{pred}}, E_{\text{pred}})}. \tag{7}$$

The higher the value of IoU, the better the algorithm would be.

2) Evaluation Metric of Position: Since our ultimate goal is to obtain the specific location of the bump, we directly evaluate the position of bumps predicted by the algorithm. The core criterion for TP is that the predicted position and the ground-truth position can match (distance between them

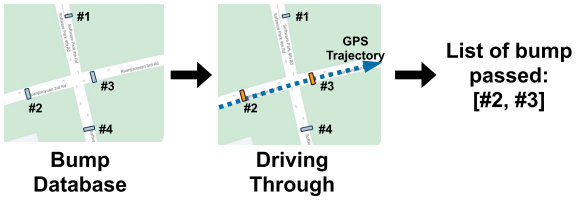


Fig. 14. We estimate the list of bumps that a trajectory has passed. The ground-truth position of bumps is stored in the database. For example, the bump list of this trajectory is [bump2, bump3].

TABLE III
PERFORMANCE OF DIFFERENT FEATURE DOMAIN

Domain	Precision	Recall	F1-score	IoU
Frequence	0.842	0.847	0.844	0.562
Time	**0.861**	**0.927**	**0.892**	0.625
Wavelet	0.860	0.813	0.835	0.526
Time+Frequence	0.834	0.856	0.844	**0.636**
All	0.850	0.876	0.862	0.607

is small). Considering that two adjacent bumps are easy to mutual interference, we set the distance threshold to 5 m and take the vehicle driving direction as the constraint. Therefore, we record longitude and latitude together with the direction when constructing a bump database. It is worth mentioning that in this case, a bump may be divided into two when it crosses lanes with opposite driving directions. Although this makes data increase, the accuracy improves.

In addition, we can calculate which bumps that a trajectory has passed through the position information (GPS). An example shows in Fig. 14.

If we know the accurate information of the bumps (trajectory pass-through), we can evaluate whether the bumps' position predicted from sensor data are correct. Therefore, we implement an algorithm to retrieve every bump in a trajectory via the bump database. The information of bumps is stored in chronological order. However, this method aims at a single trajectory. For crowdsourced data, our goal is to find out all bumps in an area through all traces. Therefore, in the experiment, we will explore the adaptability of the algorithm on crowdsourced data from an overall perspective.

D. Experiments

In this section, we design some experiments to discuss or evaluate our methods in different aspect.

1) Feature Domain: To explore more efficient features, we conduct comparative experiments on features of different domains. We extract features of each domain to train a model and evaluate them separately. Results are shown in Table III. We can find that the time domain obtains the best results. It shows that our learning target is more dependent on features in the time domain. The extraction of time-domain features has the least computationally expensive, which can show the superiority of our algorithm to some extent.

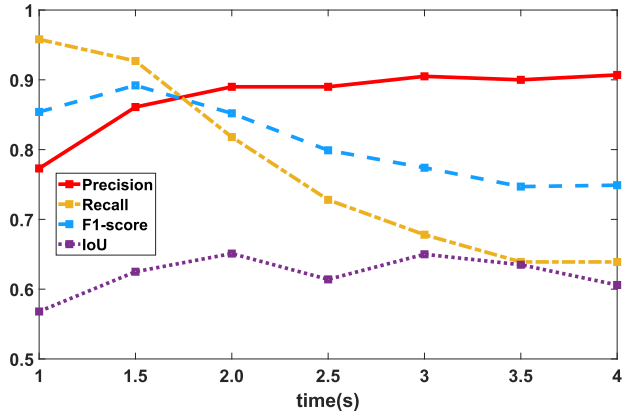


Fig. 15. Performance of different window size. The size of the window is set from 1 to 4 s.

TABLE IV
SETTINGS OF SLIDING WINDOW

	Window Size(s)	*Cover Rate**
Signal Clustering	1	0.1
Signal Recognition	1.5	0.75

* $Step = WindowSize \times (1 - CoverRate)$.

2) Sliding Window: We experimented on the performance of different window sizes. A smaller window will seize the small vibrations so that sensitively detect fluctuation signals, which leads that the model will recognize more fluctuation signals as bumps. Therefore, the precision got a worse performance. A larger window contains more information of signal obtains a better performance in precision. But this will also cause to ignore some tiny vibrations so that recall will descend. The effect of different window sizes on the accuracy of bump recognition is shown in Fig. 15.

From Fig. 15, we can find that with the increase of the window size, the precision result increases and recall decreases, which is in line with our expectations. The window size selected between 1.5 and 2 s can achieve better performance.

In our design, there are two types of sliding windows, that is, signal clustering and signal recognition. The setting of these parameters is empirical: 1) in signal clustering, a small time window is set to capture tiny vibrations, and a low cover rate to reduce the computation and 2) in signal recognition, we set a larger time window to ensure the accuracy of feature extraction and use a larger cover rate to improve the model robustness. The setting details are shown in Table IV.

3) Hidden Markov Model: To prove the robustness of HMM, we use the threshold method instead to predict whether the vehicle passes the bump. By setting a threshold, we can pick out all the fragments whose bump duration probability exceeds this threshold and regard them as the passing state. Other settings remain the same. Results are shown in Fig. 16. To achieve a high performance of the method of threshold, we should adjust the value repeatedly. Even so, its effect is still not as good as the method of HMM.

4) General Model and Personalized Model: We design experiments to evaluate the performance of the personalization model. First, we divided the dedicated datasets into three

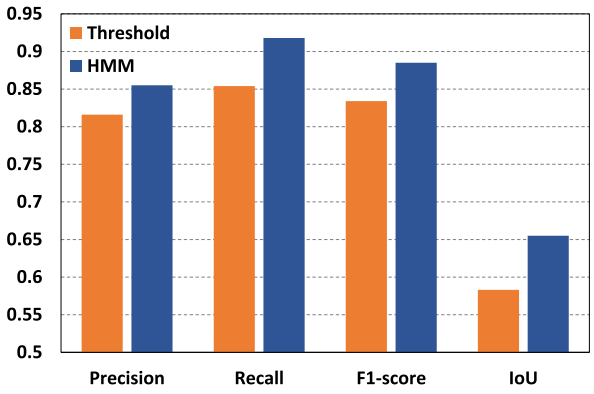


Fig. 16. Comparison of using HMM and threshold. We observe that the method of using HMM has a better performance.

TABLE V
PERSONALIZATION

	Precision	Recall	F1-score	IoU
General	0.855	0.918	0.885	0.655
Personalization*	0.847	0.937	0.889	0.681
Personalization	**0.857**	**0.938**	**0.895**	**0.685**

*** Training without general model**

parts: one for training a general model, one for processing a personalized model, and the last for testing. Furthermore, there should be a user correspondence between datasets of training personalized model and testing. These two datasets are collected by the same drivers in two areas so that one area for personalizing and the other one for testing. After training a general model, we fine-tune the model by personalized data generated by the method described in Section V for every user. Table V shows the performance of personalization.

From Table V, we conclude that the accuracy of bump recognition after personalization is better than the accuracy from the general model. In addition, we trained a personalized model without a general model, whose results are shown in the second line in the table. We found that even though its results are not necessarily better than the results of the general model, and it still gets a good performance, which fully shows that the generation of our personalized data is very useful. At the same time, combined with the further tuning of the general model, the best results were obtained, indicating the effectiveness of our method.

However, results in Table V only show the average of all users. We also want to know whether, for each user, a personalized model can be better than a general model. So, we randomly select four users and then list out their results before and after having a personalized model, as shown in Table VI.

Table VI shows that the personalized model can be accurate for each user.

5) Crowdsourced: If dedicated data is to evaluate the performance of our method, then crowdsourced one is to measure the feasibility of our system in real scenarios. The basis for evaluating crowdsourcing data is a bump database we built.

TABLE VI
PERSONALIZATION RESULT FOR INDIVIDUAL USER

	User1		User2		User3		User4	
	G	P	G	P	G	P	G	P
Precision	0.82	**0.83**	0.84	**0.85**	0.81	**0.82**	0.79	**0.82**
Recall	0.86	**0.97**	0.90	**0.95**	0.81	**0.92**	0.92	**0.93**
F1-score	0.84	**0.89**	0.87	**0.90**	0.81	**0.86**	0.85	**0.87**
IoU	0.68	**0.72**	0.65	**0.68**	0.63	**0.63**	0.64	**0.66**

G: General model
P: Personalized model

In such a database, we collect almost all bumps in an area with their GPS information include only three elements for each: longitude, latitude, and orientation. Therefore, we can evaluate the inference result by position.

First of all, we inference each trajectory separately and make statistics of their results. The inference results record in the unit of trajectory. If a trace has passed no bump, then we do not record anything for it.

Since the bumps that the trajectory theoretically pass can obtain from processing the GPS information, we compare the estimation results and the inference results. Here, we have two ways for comparison: 1) comparing by every trajectory or 2) comparing in an overall perspective. The first way can show that the accuracy of every trajectory and the second way demonstrates whether our method can find all bumps in an area. However, we found that it is unavailable. The main reason is that many drivers will deliberately bypass the edge of the speed bump to reduce vehicle turbulence, which causes no fluctuation signal in the sensor data.

To this end, we evaluate our result from a macropoint of view: whether we can find out all the bumps that should be gone through in theory by all these trajectories (Recall) and whether a bump recognized from sensor data is truly existing (Precision). For example, if we have a set of crowdsourced data, according to the GPS estimation matching, they have gone through ten bumps (repetitive bumps count as one), but bumps we found from the inertial data are 9, so the recall is regarding as 0.9. And a total of 100 bumps are inferred from inertial data, but only 70 of them can match in the database, the precision is equal to 0.7.

a) Results of crowdsourced: We randomly select 200 trajectories passing through an area and then utilize a general model to conduct inference. The precision is 0.75 and the recall is 1. The recall achieve to 1 means that all bumps are discovered by these trajectories. But the low precision means that although we found all bumps, we also identify some non-bump points as bumps. We try to analyze these wrong classification points, so we mark all the identified bumps on the actual map, as shown in Fig. 17.

From Fig. 17, we can intuitively find that the bad results can be divided into two categories: model inference error and GPS position offset. It is hard to handle the GPS offset, but improving the recognition ability of the model is feasible, which is the core role of model personalization. We train a personalized model for each trajectory and then evaluate again. The precision increases to 0.809 at this time proves that the personalized method we proposed is still effective

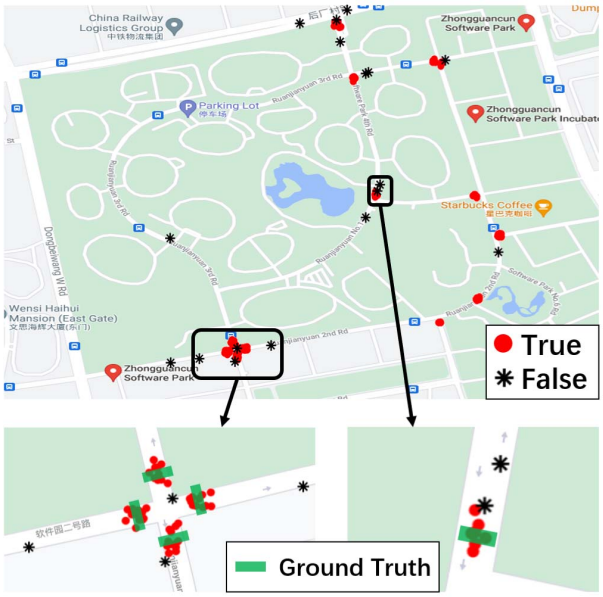


Fig. 17. Bumps recognized from crowdsourced sensor data. The red dot represents TP, which means that the predicted one has a match in database. On the contrary, the black star means an FP which has more than 5-m distance from the ground truth.

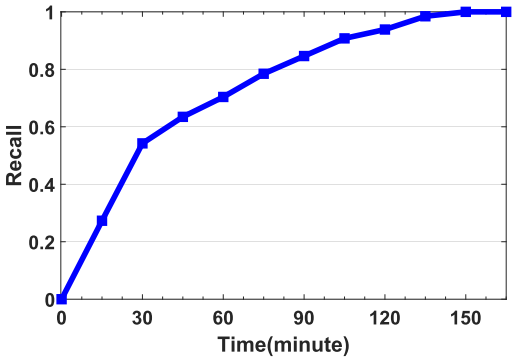


Fig. 18. Variation of recall versus time.

in crowdsourcing data. It is worth mentioning that we can also find that most of the misjudgment points are scattered separately, and the correct points cluster together, which means that we can further improve the precision by adding a simple filter.

b) Recall versus time: When we try to use crowdsourced data to find all bumps in an area, we naturally pay attention to how much data is needed to seek all the bumps. In other words, when we can stably collect the sensor data of all vehicles passing through an area within a certain period, how long will it take to find all the bumps in this area? With the help of big data on the ride-hailing platform, we experimented on the changes of recall versus time. We selected different periods for testing and averaged the results, and the change curve is shown in Fig. 18. According to statistics, about 80 online ride-hailing cars pass the selected area every hour during the working day. As shown in Fig. 18, it took about 2.5 h for the recall curve to converge to 1. But, in fact, the recall has reached 80% at 75 min. The slow rise in the second half of the curve is since some bumps are relatively off-center and few passing vehicles.

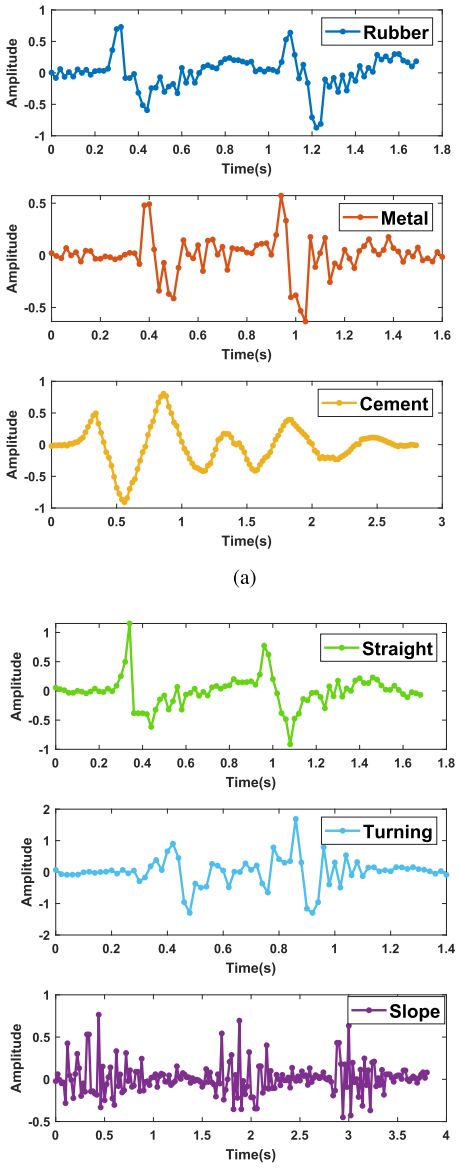


Fig. 19. Signals generated by different types of bumps. These signals collect from the same driver. (a) Signals of different materials. (b) Signals of different roads.

TABLE VII
RESULT FOR DIFFERENT TYPES OF BUMPS

Type	Recall	IoU
Rubber	0.936	0.728
Metal	0.927	0.715
Cement	1	0.837
Straight	0.932	0.727
Turning	0.915	0.674
Slope	0.864	0.666

6) Types of Bumps: In this part, we explore the impact of bump types on detection accuracy. Specifically, we illustrate the raw inertial signals by different bumps and present our recognition robustness.

TABLE VIII
TECHNICAL COMPARISON OF EXISTING WORK

	Techniques	Details	Purpose	Dataset
Pothole Patrol [3]	Threshold	Filters Parameter exhaustion	Potholes	Dedicated
Mednis [13]	Threshold	Z-Thresh Z-DIFF STDEV G-Zero	Potholes	Dedicated
Smart patrolling [7]	DTW Threshold	Filters Template matching	Potholes bumps	Dedicated
El-Walkeel [6]	Machine learning	De-Noise SVM classifier	Anomalies	Dedicated
Zheng [15]	Machine learning DTW	Random forest KNN + DTW	Potholes bumps	Dedicated
Wu [5]	Machine learning	Random forest	Potholes	Dedicated
PBS	Machine learning	XGBoost HMM Persionalization	Bumps	Dedicated Crowdsourced

TABLE IX
COMPARISON OF ACCURACY

	Precision	Recall	F1-score
Z-DIFF [13]	0.448	0.46	0.441
DTW [7]	0.844	0.573	0.712
SVM [6]	0.799	0.898	0.843
PBS	**0.857**	**0.938**	**0.895**

Fig. 19 shows the raw signals by different bumps. In Fig. 19(a), we can find that rubber and metal bumps are similar in signal shape as cars pass. When driving through cement bumps with poor shock absorption, vehicles go much slower, and there are more pronounced fluctuations in the signal. Therefore, our model achieves perfect recognition accuracy at cement bumps. In Fig. 19(b), we observe that a bump on straight roads usually causes two signal fluctuations by front and rear wheels, while a bump at turning causes continuous vibrations. In addition, the bumps on downhill slopes are the most special with long-term slight shakes.

Table VII shows the detection accuracy on different types of bumps. Recall stands for accuracy against missed bumps, which assesses the model's robustness to identify different types of bumps. The results have shown that our model can effectively adapt to different types of bumps, on both recall and IoU.

E. Comparison

We compare our system with some existing works. Although there has been a lot of work to detect road anomalies, there are differences in the datasets and evaluation criteria, so it is of little significance to compare directly according to the results in the articles. We list some classical works with their details in Table VIII. Since most of the work has no open-source code and dataset, we can only try to reproduce some methods in their articles and use our data and evaluation criteria for evaluation. A comparison of accuracy results is shown in Table IX. We can find that the threshold gets the worst performance, even though we try sets of parameters. The DTW depends on the selection of the pattern and the segmentation data. Furthermore, we use SVM for binary classification. Although the result is not bad, it is still not as good as our PBS.

VIII. CONCLUSION

In this article, we identify road bumps by crowdsourced smartphones inside vehicles. Specifically, we design a novel signal recognition method to eliminate the diversity of bump duration during driving. We also explore a personalized training and inference mechanism to extract individual's bump signal and record their locations on the map. Extensive experiments in large-scale real-world datasets have demonstrated our effectiveness.

REFERENCES

[1] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, and G. Luo, "Smartphone-based real time vehicle tracking in indoor parking structures," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 2023–2036, Jul. 2017.

[2] A. Motroni, A. Buffi, P. Nepa, and B. Tellini, "Sensor-fusion and tracking method for indoor vehicles with low-density UHF-RFID tags," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–14, 2021.

[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*, Jun. 2008, pp. 29–39.

[4] A. Troiano, E. Pasero, and L. Mesin, "New system for detecting road ice formation," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 3, pp. 1091–1101, Mar. 2011.

[5] C. Wu *et al.*, "An automated machine-learning approach for road pothole detection using smartphone sensor data," *Sensors*, vol. 20, no. 19, p. 5564, Sep. 2020.

[6] A. S. El-Wakeel, J. Li, A. Noureldin, H. S. Hassanein, and N. Zorba, "Towards a practical crowdsensing system for road surface conditions monitoring," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4672–4685, Dec. 2018.

[7] G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, "Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing," *Pervasive Mobile Comput.*, vol. 40, pp. 71–88, Sep. 2017.

[8] S. Sattar, S. Li, and M. Chapman, "Road surface monitoring using smartphone sensors: A review," *Sensors*, vol. 18, no. 11, p. 3845, Nov. 2018.

[9] I. Schiopu, J. P. Saarinen, L. Kettunen, and I. Tabus, "Pothole detection and tracking in car video sequence," in *Proc. 39th Int. Conf. Telecommun. Signal Process. (TSP)*, Jun. 2016, pp. 701–706.

[10] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic pavement crack detection based on structured prediction with the convolutional neural network," 2018, *arXiv:1802.02208*.

[11] M. Quintana, J. Torres, and J. M. Menéndez, "A simplified computer vision system for road surface inspection and maintenance," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 608–619, Mar. 2016.

[12] D. Arya *et al.*, "Transfer learning-based road damage detection for multiple countries," 2020, *arXiv:2008.13101*.

[13] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSS)*, Jun. 2011, pp. 1–6.

[14] A. Mohamed *et al.*, "RoadMonitor: An intelligent road surface condition monitoring system," in *Intelligent Systems*. Cham, Switzerland: Springer, 2015, pp. 377–387.

[15] Z. Zheng *et al.*, "A fused method of machine learning and dynamic time warping for road anomalies detection," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–13, 2020.

[16] M. Perttunen *et al.*, "Distributed road surface condition monitoring using mobile phones," in *Proc. Int. Conf. Ubiquitous Intell. Comput.* Berlin, Germany: Springer, 2011.

[17] V. K. Nguyen, É. Renault, and V. H. Ha, "Road anomaly detection using smartphone: A brief analysis," in *Proc. Int. Conf. Mobile, Secure, Program. Netw.* Cham, Switzerland: Springer, 2018.

[18] F. Seraj *et al.*, "RoADS: A road pavement monitoring system for anomaly detection using smart phones," in *Big Data Analytics in the Socialand Ubiquitous Context*. Cham, Switzerland: Springer, 2015, pp. 128–146.

[19] D. N. Joanes and C. A. Gill, "Comparing measures of sample skewness and kurtosis," *J. Roy. Stat. Soc.*, vol. 47, no. 1, pp. 183–189, 1998.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[21] T. Chen *et al.*, "Xgboost: Extreme gradient boosting," R Package Version 0.4-2 1.4, 2015, pp. 1–4.

[22] Y. Yuan and X. Che, "Research on road condition detection based on crowdsensing," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2019, pp. 804–811.

[23] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. KDD Workshop*, 1994, vol. 10, no. 16, 1994, pp. 359–370.

[24] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: The dtw package," *J. Stat. Softw.*, vol. 31, no. 7, pp. 1–24, 2009.

[25] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. MAG-3, no. 1, pp. 4–16, Jan. 1986.

[26] A. Rosebrock. (2016). *Intersection Over Union (IOU) for Object Detection*. [Online]. Available: http://www.pyimagesearch.com/2016/11/07/intersection-overunion-iou-for-objectdetection

Xuan Xiao received the B.S. degree in network engineering from the China University of Mining and Technology, Xuzhou, China, in 2016. He is currently pursuing the Ph.D. degree in software engineering with Beijing Jiaotong University, Beijing, China.

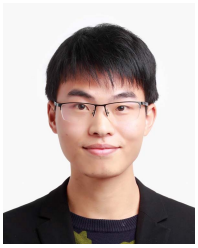
His research interests include mobile computing and applications and the Internet of Things.

Ruipeng Gao received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010, and the Ph.D. degree from Peking University, Beijing, in 2016.

He was a Visiting Scholar with Purdue University, West Lafayette, IN, USA, in 2019. He is currently an Associate Professor with the School of Software Engineering, Beijing Jiaotong University, Beijing. His research interests include mobile computing and applications, the Internet of Things, and intelligent transportation systems.

Weiwei Xing (Member, IEEE) received the B.S. degree in computer science and technology and the Ph.D. degree in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2001 and 2006, respectively.

She was a Visiting Scholar with the University of Pennsylvania, Philadelphia, PA, USA, from 2011 to 2012. She is currently a Professor with the School of Software Engineering, Beijing Jiaotong University. Her research interests include computer vision, pattern recognition, and intelligent transportation algorithms and applications.



Chi Li received the B.S. degree from Nanchang Hangkong University, Nanchang, China, in 2014, and the M.S. degree from Beihang University, Beijing, China, in 2017.

He is currently a Location Algorithm Engineer with DiDi Company, Beijing. His research interests include inertial navigation, integrated navigation, and machine learning.

Lei Liu received the B.S. degree from Shandong University, Jinan, China, in 2010, and the M.S. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2013.

He is currently an Expert Algorithm Engineer with DiDi Company, Beijing, and in charge of positioning and POI recommendation.