BikeMate: Bike Riding Behavior Monitoring with Smartphones

Weixi Gu
Tsinghua-Berkeley Shenzhen Institute
Tsinghua University
guweixigavin@gmail.com

Zimu Zhou
Computer Engineering and Networks Laboratory
ETH, Zurich
zzhou@tik.ee.ethz.ch

Yuxun Zhou
Electrical Engineering and Computer Science
University of California, Berkeley
yxzhou@berkeley.edu

Han Zou
Electrical Engineering and Computer Science
University of California, Berkeley
hanzou@berkeley.edu

Yunxin Liu
Microsoft Research Asia
yunxin.liu@microsoft.com

Costas J. Spanos
Electrical Engineering and Computer Science
University of California, Berkeley
spanos@eecs.berkeley.edu

Lin Zhang
Tsinghua-Berkeley Shenzhen Institute
Tsinghua University
linzhang@tsinghua.edu.cn

ABSTRACT

Detecting dangerous riding behaviors is of great importance to improve bicycling safety. Existing bike safety precautionary measures rely on dedicated infrastructures that incur high installation costs. In this work, we propose BikeMate, a ubiquitous bicycling behavior monitoring system with smartphones. BikeMate invokes smartphone sensors to infer dangerous riding behaviors including lane weaving, standing pedalling and wrong-way riding. For easy adoption, BikeMate leverages transfer learning to reduce the overhead of training models for different users, and applies crowdsourcing to infer legal riding directions without prior knowledge. Experiments with 12 participants show that BikeMate achieves an overall accuracy of 86.8% for lane weaving and standing pedalling detection, and yields a detection accuracy of 90% for wrong-way riding using crowdsourced GPS traces.

CCS CONCEPTS

• **Human-centered computing** → *Smartphones*; *Personal digital assistants*;

KEYWORDS

Bike; Smartphones; Activity Recognition

ACM Reference format:
Weixi Gu, Zimu Zhou, Yuxun Zhou, Han Zou, Yunxin Liu, Costas J. Spanos, and Lin Zhang. 2017. BikeMate: Bike Riding Behavior Monitoring with Smartphones. In *Proceedings of 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Melbourne, Australia, November 2017 (MobiQuitous 2017),* 10 pages.
https://doi.org/10.1145/3144457.3144462

MobiQuitous 2017, November 2017, Melbourne, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144462

1 INTRODUCTION

Due to the environmental and health benefits, bicycles continue to gain popularity as a sustainable transportation alternative. There is a rapid growth of bike sharing programs across major cities worldwide. According to a report by Roland Berger [1], over 1,000 bike sharing systems are already in operation and the market is expected to grow by 20% every year by 2020. However, cycling safety can be easily overlooked by many bicyclists. National Highway Traffic Safety Association (NHTSA) report that there were 726 killed and an additional 50,000 injured in the US due to traffic accidents that involves cyclists in 2014 alone [25]. Dangerous riding behaviors are one of the main causes of such tragedies. For instance, lane weaving and standing pedaling sometimes provoke balancing difficulties and risk falling from the bike. Wrong-way riding *i.e.*, cycling against the direction of legal traffic, usually causes head-on collision or traffic congestion. Therefore, it is necessary to detect dangerous cycling behaviors to alert the bicyclists in time to avoid potential accidents.

Many bike safety systems have been developed to improve the visibility of bikes [9] or warn rear-approaching vehicles [28] with extra infrastructures. The high installation cost prevents their large-scale deployment, especially for bike sharing systems that make profits. The emergence of smartphones provides a ubiquitous opportunity to deliver mobile safety services. Pioneer works [4][6][19] have exploited the built-in sensors in smartphones to monitor driving and riding behaviors. For instance, Chen *et al.* [4] develop a middleware to detect abnormal driving behaviors such as weaving and fast U-turns. Johnson *et al.* [19] distinguish normal and aggressive driving behaviors with inertial sensors. Condro *et al.* [6] detect high-risk riding maneuvers of motorcyclists using smartphones. However, those solutions cannot be directly applied in dangerous bicycle behavior detection due to the difference in riding and driving behaviors. For example, standing pedalling is unique in cycling,

and bicyclists are more likely to ride in the wrong directions than motorcyclists and drivers.

With the rapid development of sensor technologies embedded in the smartphones, many researchers utilize it as a sensing platform to study individual behaviours [16, 31]. In this paper, we take this intuition and propose BikeMate based on our previous work [14], a pervasive bicycle riding behavior monitoring system with smartphones. It evokes the embedded accelerometer and gyroscope of smartphones to monitor the motions of bicyclists and employs the GPS to track their riding directions. BikeMate then detects dangerous riding behaviors including lane weaving, standing pedalling and wrong-way riding. Towards a ubiquitous solution, BikeMate focuses on addressing the following challenges.

- How to robustly identify the distinctive patterns of dangerous bicycling behaviors from noisy inertial measurements?
- How to effectively train dangerous bicycling behavior models for different users with minimal efforts?
- How to determine the legal riding directions of bike-ways from GPS trajectories without prior knowledge?

Contributions. BikeMate addresses the above challenges by *(i)* extracting effective features from each kind of sensory measurements (Sec. 4.1), *(ii)* adopting a transfer kernel learning method to share information among models for different bicyclists (Sec. 4.2), and *(iii)* designing a crowdsourcing scheme to learn the legal riding directions (Sec. 4.3). Evaluations with 12 volunteers over two weeks show that BikeMate achieves an overall accuracy of 86.8% in riding behavior detection (lane weaving, standing pedalling and normal riding) and an accuracy of over 90% in wrong-way riding detection using crowdsourced GPS traces.

The rest of the paper clarifies each of the above contributions, beginning with a literature review on related works, followed by the detailed design, implementation, and evaluation of BikeMate.

2 RELATED WORK

BikeMate is related to the following categories of research.

Bike Safety Infrastructures. In addition to building bike lanes and enforcing wearing helmets, many infrastructures have been designed to improve cycling safety. The Cyber-Physical Bicycle system [28] automatically detects rear-approaching vehicles and reminds the bicyclist using bike-mounted cameras for real-time video processing. With large-scale cyclist data collected from road cameras, Sayed *et al.* [27] propose an automatic bike safety diagnosis via traffic conflict analysis. Smart Flashlight [7] installs a projector and a smartphone on bikes to project maps on the road for nighttime bike navigation. Krauter *et al.* [21] propose to enhance the safe communication among group cyclists using gesture recognition and LEDs sewed into the shirts of the cyclists.

Our work is complementary to this thread of research in *(i)* improving riding safety without extra hardware installed on bikes, and *(ii)* detecting dangerous riding behaviors of cyclists.

Smartphone-based Driving behavior Monitoring. There has been growing research in leveraging smartphones for human behavior pattern detection [12, 15], especially in the traffic area [13]. Chu *et al.* [5] utilize inertial sensors in smartphones to recognize micro-activities to distinguish passengers and drivers. CarSafe [32] detects lane changes and drowsy drivers based on the photos of both the driver and the environment outside captured by the dual cameras on smartphones. D^3 [4] identifies fine-grained abnormal driving behaviors including weaving, swerving, side-slipping, fast U-turn, wide-radius turn and sudden braking with smartphone sensors. V-Sense [3] monitors vehicle steering and differs lane-changes, turns, and driving on curvy roads using a similar approach. The most relevant work is [6], where acceleration and GPS traces are used to detect high-risk motorcycle maneuvers or accidents.

Our work is inspired by these research efforts. However, dangerous cycling behavior detection can be more complex than driving behavior monitoring [6].

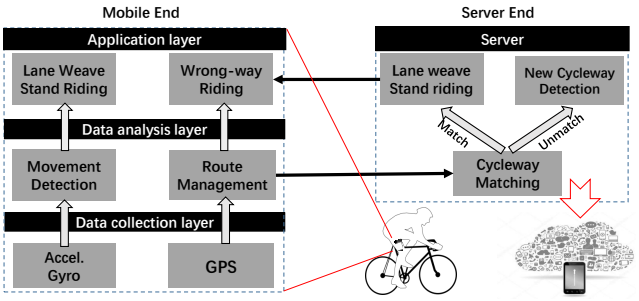

Figure 1: Architecture of BikeMate. It identifies lane weaving and standing pedalling from phone inertial measurements and detects wrong-way riding from crowdsourced GPS traces.

3 SYSTEM OVERVIEW

This section presents the overview of our BikeMate design.

3.1 Scope

We focus on three dangerous bike riding behaviors including lane weaving, standing pedalling and wrong-way riding, which are defined as follows.

- **Lane Weaving.** Ride alternatively from one side of the bike lane to the other, *i.e.*, in a S-shape.
- **Standing Pedalling.** Stand up to pedal, usually to accelerate, but may lose balance.
- **Wrong-way Riding.** Ride in the opposite direction against the direction of traffic or the wrong side of the road.

Fig. 7 illustrates the above three dangerous biking riding behaviors.

As a ubiquitous bicycle riding behavior monitoring system, BikeMate needs to meet two requirements.

- Accuracy. As a service to improve safety, highly accurate detection of dangerous riding behaviors is important to avoid potential accidents and not to distract the attention of bicyclists.
- Usability. Since the riding behaviors may differ from person to person, it is beneficial to reduce the overhead on per-person training to build riding behavior models for each individual. BikeMate should also work with minimal prior knowledge because many maps lack detailed information for bike lanes.

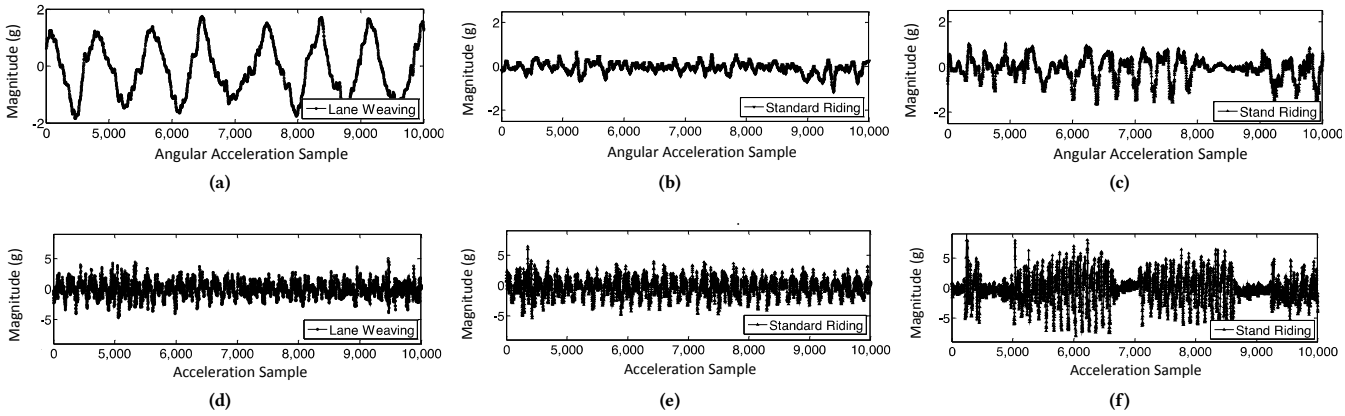

Figure 2: The angular acceleration patterns of (a) lane weaving, (b) standard riding and (c) standing pedalling collected from a gyroscope and the overall acceleration patterns of (d) lane weaving, (e) standard riding and (f) standing pedalling collected from an accelerometer.

3.2 Work Flow

Fig. 1 shows the architecture of BikeMate. It consists of a (mobile) client end and a server end. The client end employs smartphone sensors to track the movements of cyclists and to record their GPS information at the **data collection layer**. These measurements are then processed by the **data analysis layer** to detect dangerous riding behaviors including lane weaving, standing pedalling and wrong-way riding. The **application layer** reminds the cyclist via predefined alarm mechanisms.

3.2.1 Data Collection Layer. We assume a BikeMate user carries a smartphone while riding a bike. The data collection layer evokes the built-in accelerometer to record 3-axis acceleration and the gyroscope to measure angular acceleration data. It also turns on the phone GPS to track the routes of the user. The collected raw data are then delivered into the data analysis layer.

3.2.2 Data Analysis Layer. The data analysis layer consists of a **riding movement detection module** and a **riding route management module**. In the riding movement detection module, BikeMate identifies lane weaving and the standing pedalling from inertial sensors. It also applies transfer learning techniques to reduce the overhead when training models for different users. The riding route management module is responsible for detecting wrong-way riding using crowdsourced GPS traces.

3.2.3 Application Layer. The application layer runs at the backend in usual for power saving, and reminds cyclists of their dangerous riding behaviors detected by the data analysis layer. Once a dangerous riding event is detected, BikeMate delivers vibration and sound to the user.

4 BIKEMATE DESIGN

This section presents the technical details of BikeMate.

4.1 Detecting Lane Weaving and Standing Pedalling

BikeMate distinguishes lane weaving, standing pedalling and normal riding using phone accelerometer and gyroscope.

4.1.1 Observations. Intuitively, lane weaving alternatively increases the forces of the left and the right sides, thus resulting in large angular acceleration values. For standing riding, the legs of the rider tend to move in a wider range than for normal riding, which leads to larger acceleration readings.

Fig. 2 shows the acceleration and the angular acceleration readings collected by a phone accelerometer and a phone gyroscope. We observe that the magnitude of the angular acceleration for lane weaving is notably greater than the acceleration in standing pedalling and normal riding. The fluctuations of the angular acceleration also follow an S-shape. During standing pedalling, the overall acceleration is consistently larger than that in lane weaving and normal riding. Here we calculate the overall acceleration because the accelerometer can be placed in arbitrary locations and its coordinate system may vary. However, the overall acceleration will still capture the large acceleration induced by the wide leg moving range during standing pedalling. Normal riding means riding smoothly with few fluctuations. Hence both the overall acceleration and the angular acceleration readings are relatively stable over time.

4.1.2 Feature Extraction and Classification. Based on the above observations, we adopt five features to capture the distinctive patterns in the overall acceleration and the angular acceleration readings including average magnitude, standard deviation, average absolute difference, binned distribution and period of riding. The definitions of the five features are listed as follows, where a_j is the rooted square of one sample of the overall acceleration measured by the phone accelerometer or the angular acceleration measured by the phone gyroscope.

- Average Magnitude (AM):

$$AM = \sum_{j=1}^{N} a_j \tag{1}$$

- Standard Deviation (SD):

$$SD = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(a_j - \mu)^2} \tag{2}$$

- Average Absolute Difference (AAD):

$$AAD = \frac{1}{N}\sum_{j=1}^{N}\left|a_j - \mu\right| \tag{3}$$

- Binned Distribution (BD):

$$BD = \sum_{i=1}^{K}\sum_{j=1}^{N} sign\left[a_j - a_{min} + \frac{(i-1)}{K}(a_{max} - a_{min})\right] \left[a_{min} + \frac{i}{K}(a_{max} - a_{min}) - a_j\right]. \tag{4}$$

- Period of Riding (PoR):

$$PoR = arg\ max_{\tau} \frac{\sum_{k=1}^{\tau-1}[a_{m+k} - \mu(m,\tau)][a_{m+\tau+k} - \mu(m+\tau,\tau)]}{\tau\sigma(m,\tau)\sigma(m+\tau,\tau)} \tag{5}$$

The parameters μ, a_{min} and a_{max} in Eq.1, 2, 3, and 4 stand for the average, the minimal and maximal samples of the sliding window with N samples. K in Eq.4 is a preset range of bins divided by $(a_{max} - a_{min})$, which is empirically set as 10. $\mu(m,\tau)$, $\sigma(m,\tau)$ and τ in Eq.5 indicate to the mean, standard deviation and the period of a acceleration sequence starting from m point. BikeMate sets the sampling rate of the accelerometer and the gyroscope as 100 Hz, and adopts a 15-second sliding window to extract features. The sampling rate and the window size are suitable to capture the riding movement based on our experimental results.

For both the overall acceleration and the angular acceleration, we extract a feature vector $X =< X_{AM}, X_{SD}, X_{AAD}, X_{BD}, X_{PoR} >$ and combine them as the input for riding behavior classification. Then we adopt a support vector machine (SVM) [29] to identify the riding behavior $Y = \{Y_{LW}, Y_{SP}, Y_{NR}\}$, where LW, SP and NR represent lane weaving, standing pedalling, and normal riding, respectively. We choose SVM for its simplicity and effectiveness in differentiating the three riding behaviors.

4.2 Transferring to Different Users

Due to the complexity and diversity riding behaviors, it is necessary to train a user-specific riding behavior classifier for each BikeMate user. Fig. 3 illustrates the traces of the overall acceleration of three different users when they ride bikes normally. We observe that the acceleration periods of user 3 is much longer than that of user 1 and user 2. This is because user 1 and user 2 ride much faster than user 3, while user 3 alternatively speeds up and relaxes.

However, it is labour-intensive to build each user-dependent riding behavior classifier from scratch. Since the differences among the three riding behaviors are relatively generic, it is possible to capture these inherent characteristics and share among different users so as to speed up the training process of user-specific riding

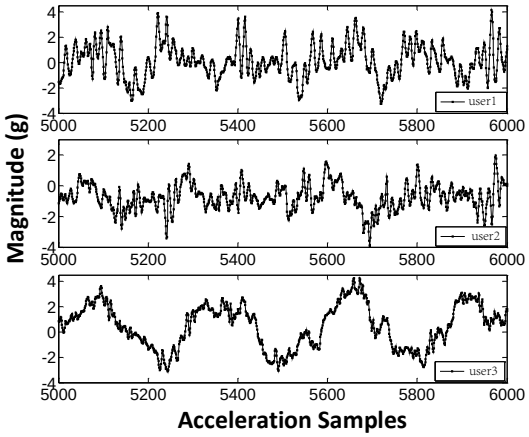

Figure 3: The overall acceleration traces of three different users during normal riding.

behavior models. For instance, comparing the acceleration traces of user 1 and user 2 from Fig. 3, the general trends resemble each other even though there are still individual differences.

To reduce the overhead to train classifiers for each individual, we adopt a transfer kernel learning (TKL) method [23]. It uncovers the latent features that are invariant among different users, and transfers the similarities between the trained users and the test users to compensate for the disparities among individuals.

The transfer kernel learning method applied in BikeMate is based on the framework of the Maximum Mean Discrepancy (MMD), which compares data distributions based on the distance between the means of samples from the two domains in the Reproducing Kernel Hilbert Space (RKHS). In BikeMate, we adopt a more flexible approximation criterion to better match the source domain and the target domain. Specifically, it formalizes the distribution discrepancy between the extrapolated source kernel and the target source kernel by the Nyström approximation error [22]. A family of spectral kernels is designed by extrapolating the target eigensystem on source data using the Mercer's theorem [24]. Then the spectral kernel that minimizes the approximation error to the ground truth source kernel and that has the most similar feature-space distribution is selected as the input data for the training procedure. Therefore, the learned domain-invariant kernel can respect both the target eigensystem and source approximation quality [23].

In general, given the labeled features $\mathcal{X}_L$ and the unlabeled features $\mathcal{X}_U$, the first step of TKL is to figure out the shared similarities between the two feature domains via Nyström approximation under the MMD framework. The output of the first step is a new domain-invariant kernel, which is used as the input of SVM. Finally, we combine the trained model and the transitional kernel to build an adaptive and transferable riding behavior model for a new BikeMate user even if his/her measurement data distribution differs from those used in the training phase. We elaborate on the details of the TKL method for our riding behavior classification as follows.

(1) TKL computes the labelled acceleration kernel $\mathbf{K}_L$ and the unlabelled acceleration kernel $\mathbf{K}_U$ using a preset input kernel function k, *e.g.*, RBF kernel $k(\mathbf{X}_L, \mathbf{X}_U) = e^{-\gamma \|\mathbf{X}_L - \mathbf{X}_U\|^2}$. Meanwhile, it calculates the cross-domain kernel matrix $\mathbf{K}_{LU}$ of the labelled $\mathbf{X}_L$ and unlabelled $\mathbf{X}_U$ acceleration features.

(2) TKL evaluates the differences of distribution between the labelled features and the unlabelled features in the kernel Hilbert space. As such, TKL first builds an extrapolated source kernel $\overline{\mathbf{K}}_L \in \mathbb{R}^{l_L \times l_L}$ by the eignesystem $\{\mathbf{\Phi}_U, \mathbf{\Lambda}_U\}$ of the target kernel $\mathbf{K}_U$ based on Nyström approximation.

(3) The target eigenvector matrix $\mathbf{\Phi}_U$ and target eigenvalue matrix $\mathbf{\Lambda}_U$ can be easily obtained through solving the standard eigenvalue problem by Eq. 6.

$$\mathbf{K}_U \mathbf{\Phi}_U = \mathbf{\Phi}_U \mathbf{\Lambda}_U \tag{6}$$

With the calculated eignesystem of $\mathbf{K}_U$ and the cross-domain kernel matrix $\mathbf{K}_{LU}$, the eigenvector matrix of the extrapolated source kernel $\overline{\mathbf{K}}_U$ is derived using the Mercer theorem in Eq. 7

$$\overline{\mathbf{\Phi}}_L \simeq \mathbf{K}_{LU} \mathbf{\Phi}_U \mathbf{\Lambda}_U^{-1}. \tag{7}$$

(4) According to the Nyström approximation error, a modified labelled kernel $\overline{\mathbf{K}}_L$ can be built by minimizing the distribution divergence, *i.e.*, the approximation error, which is computed as Eq. 8:

$$\begin{aligned} \min_{\mathbf{\Lambda}} \|\overline{\mathbf{K}}_L - \mathbf{K}_L\|^2 = \|\overline{\mathbf{\Phi}}_L \mathbf{\Lambda} \overline{\mathbf{\Phi}}_L^\top - \mathbf{K}_L\|^2, \\ \lambda_i \geq \zeta \lambda_{i+1}, i = 1, \ldots, l_U - 1, \\ \lambda_i \geq 0, i = 1, \ldots, l_U, \end{aligned} \tag{8}$$

where $\mathbf{\Lambda} = diag\{\lambda_1, \ldots, \lambda_{l_U}\}$ are the l_U non-negative eigenspectrum parameters and ζ is the eigenspectrum damping factor. The eigenspectrum parameters $\mathbf{\Lambda}$ can be estimated by translating the optimization problem (8) into a convex quadratic programming (QP) [26] with linear constraints. Eq. 8 can be reformulated as

$$\begin{aligned} \min_{\lambda} \lambda^\top \mathbf{Q} \lambda - 2\mathbf{r}^\top \lambda \\ \mathbf{C}\lambda \geq 0 \\ \lambda \geq 0 \end{aligned} \tag{9}$$

where the parameters $\lambda = (\lambda_1, \ldots, \lambda_{l_U})$, $\mathbf{Q} = (\overline{\mathbf{\Phi}}_\mathbf{L}^\top \overline{\mathbf{\Phi}}_\mathbf{L}) \odot (\overline{\mathbf{\Phi}}_\mathbf{L}^\top \overline{\mathbf{\Phi}}_\mathbf{L})$ and $\mathbf{r} = \mathbf{diag}(\overline{\mathbf{\Phi}}_\mathbf{L}^\top \mathbf{K}_\mathbf{L} \overline{\mathbf{\Phi}}_\mathbf{L})$ are the QP coefficient matrices, and $\mathbf{C} = \mathbf{I} - \zeta \bar{\mathbf{I}}$ is the constraint matrix. We solve the QP problem using the interior-point algorithm.

(5) After estimating the optimal eigenspectrum parameters $\mathbf{\Lambda}$, TKL builds up the domain-invariant kernel $\overline{\mathbf{K}}_\mathcal{A}$ on both the source and target data $\mathcal{A} = \mathcal{L} \cup \mathcal{U}$. $\overline{\mathbf{K}}_\mathcal{A}$ is constructed from the domain-invariant eigensystem $\{\mathbf{\Lambda}, \overline{\mathbf{\Phi}}_\mathcal{A}\}$ according to the spectral kernel design as

$$\overline{\mathbf{K}}_\mathcal{A} = \overline{\mathbf{\Phi}}_\mathcal{A} \mathbf{\Lambda} \overline{\mathbf{\Phi}}_\mathcal{A}^{-1}, \tag{10}$$

where $\overline{\mathbf{\Phi}}_\mathcal{A} \doteq [\overline{\mathbf{\Phi}}_L; \mathbf{\Phi}_U]$.

(6) The estimated domain-invariant kernel $\overline{\mathbf{K}}_\mathcal{A}$ can be adopted as the input of an SVM to construct the adaptive localization model via the LIBSVM package. The output of the model are the predicted riding movements of the new cyclist without labels $\boldsymbol{\ell}_u$.

Algorithm 1 illustrates the process of the transfer kernel learning.

Algorithm 1: Transfer Kernel Learning of Riding behaviors

Input: $\mathcal{X}_L = \{\mathbf{X}_{Li}\}_{i=1}^N$: labeled features collected from the calibrated riders with N instance;
$\mathcal{X}_U = \{(\mathbf{X}_{Ui}\}_{i=1}^M$: unlabeled features collected from the uncalibrated riders with M instance;
$\mathcal{Y}_L = \{\mathrm{Y}_{Li}\}_{i=1}^N$: labeled events collected from the calibrated riders with N instance;
k: kernel type; ζ: eigenspectrum damping factor
1) Compute matrices $\mathbf{K}_L$, $\mathbf{K}_U$ and $\mathbf{K}_{LU}$ using kernel function k
2) Eigendecompose $\mathbf{K}_U$ to obtain the unlabelled eigenvector matrix $\mathbf{\Phi}_U$ and unlabelled eigenvalue matrix $\mathbf{\Lambda}_U$ using Eq. (6)
3) Compute the eigenvector matrix of the extrapolated labelled kernel $\overline{\mathbf{K}}_L$ by Eq.(7)
4) Minimize the distribution divergence between the extrapolates labelled kernel $\overline{\mathbf{K}}_L$ and the unlabelled kernel $\mathbf{K}_U$ and solve the QP problem for eigenspectrum $\mathbf{\Lambda} = diag\{\lambda_1, \ldots, \lambda_{l_U}\}$
5) Compute the domain-invariant kernel matrix $\overline{\mathbf{K}}_\mathcal{A}$ using Eq.(10)
6) Inferring the riding movement $\mathbf{Y}_U$ using SVM with the domain-invariant kernel matrix $\overline{\mathbf{K}}_\mathcal{A}$ via LIBSVM package.
Output $\boldsymbol{\ell}_U$: inferred riding movements

4.3 Detecting Wrong-way Riding

BikeMate detects wrong-way riding from crowdsourced GPS traces. Although many commercial map databases store large amounts of GPS trajectories, the legal directions of two-side bike lanes are often missing. Therefore in BikeMate, we propose to construct a bike-way route database and label the legal riding direction of each bike-way via crowdsourcing. Fig. 4 details the work flow of the wrong-way riding detection. The raw GPS trace is first pre-processed and then matched with the trajectories and compare with the legal direction stored in the GPS trajectory database at the server end. If the GPS trace cannot be matched, it will be stored as candidates to generate new bike lanes. As next, we present the details of each step.

4.3.1 Pre-processing. The pre-processing of the raw GPS traces consists of two steps.

Noise Elimination. The GPS signals are easily interrupted by the surroundings (*e.g.*, buildings), and thus the GPS samples are usually noisy. Apart from the latitude and the longitude, a standard GPS sample also contains an error radius, which measures the realm of the bicyclist's true position. A large error radius indicates low confidence of the geographic position reported by the GPS. Accordingly, BikeMate regards the GPS samples whose error radius is larger than 20m as noise.

Smoothing. In the second step, BikeMate leverages the Weighted Moving Average (WMA) [18] to smooth the GPS trace as follows.

$$WMA_M = \frac{\sum_{k=1}^{n} (k P_{M-n+k})}{\sum_{k=1}^{n} k} \tag{11}$$

In BikeMate, we empirically select a five-point WMA to smooth the GPS trace. Fig. 5 shows the GPS traces before and after pre-processing. We observe that the GPS traces naturally cluster into two groups, indicating two bike-ways.

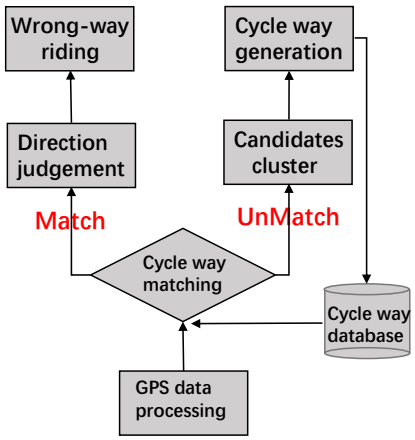

Figure 4: Work flow of the wrong-way riding detection. BikeMate first constructs a bike-way (cycle-way) database that automatically labels the legal riding directions and then identifies wrong-way riding based on the crowdsourced database.

4.3.2 GPS Trace Matching. After pre-processing, BikeMate tries to match the GPS traces with the GPS trajectories stored in the server via Dynamic Time Warping (DTW) [2]. DTW is a dynamic programming based similarity measure for sequences which may vary in time or speed. In DTW, the two sequences are first reconstructed by non-linear "warping" in the time domain to compare their similarity independent of non-linear temporal variations. Therefore, DTW based trace matching can be well applied in the GPS trace matching.

Given two GPS traces A and B with lengths of M and N samples, DTW first constructs a distance matrix $d[M \times N]$ as Eq. 12

$$d(i,j) = (a_i - b_j)^2 \tag{12}$$

where a_i and b_j are the ith and jth elements in A and B, respectively.

Taking $d[M \times N]$ as input, DTW returns a warping path $P = \{p_1, p_2, p_3, \ldots, p_k\}$, where $p_i = (x, y) \in [1 : M] \times [1 : N]$ for $i \in [1 : k]$.

Fig. 6 illustrates the matching process. To generate the warping path, DTW constructs a cost matrix $C[M \times N]$, which stands for the minimum cost to reach any point (i, j) in the matrix from $(1, 1)$ in a dynamic programming fashion. For instance, (i, j) can be reached from $(i-1, j-1)$, $(i, j-1)$ and $(i-1, j)$. The algorithm picks the one with minimum cost as follows.

$$C(i,j) = d(i,j) + min(C(i-1,j-1), C(i,j-1), C(i-1,j)) \tag{13}$$

If the smallest cost between the GPS trace and those in the database is smaller than a threshold, we regard it as a match. The preset threshold is empirically set as 15 based on our experiments.

If matched, BikeMate determines the rider's current position and then compares his/her riding direction with the legal direction of the bike-way. BikeMate computes a rider's riding direction by partitioning his/her GPS trace into disjointed grids ($5m \times 5m$). Assuming a GPS segment with a start coordinate (X_1, Y_1) and an end coordinate (X_2, Y_2), the riding direction $\tau \in [-90, 90]$ is:

$$\tau = arctan(\frac{Y_2 - Y_1}{X_2 - X_1}) \tag{14}$$

The two red arrows in Fig. 5 show the legal directions of bike lanes calculated by the method above.

If unmatched, BikeMate regards this GPS trace as a new bike lane candidate, and generates a new bike lane by crowdsourcing.

4.3.3 New Route Generation. For the GPS traces that cannot be matched with those stored in the database, BikeMate tries to generate a new route. The process consists of two main steps.

Trace Clustering. BikeMate adopts the single linkage clustering algorithm [11] to cluster the smoothed GPS traces using the Hausdorff distance [8]. Hausdorff distance only considers the longest distance between two bike-ways to mitigate the impact of the noise within a distance threshold. Based on our observation, we set the distance threshold as 10m.

Centerline Fitting. Given a cluster, if the number of trajectories in one cluster excesses λ, BikeMate regards it as a bike lane, and then extracts its centerline. λ is set as 3 based on the experimental results. The polygonal principal curve algorithm [20] is applied to localize the centerline. This algorithm generates the centerline by minimizing its mean square error of the GPS samples of the GPS trajectories within the same cluster. Compared with other algorithms, the polygonal principal curve algorithm is able to generate the centerline with onle a few GPS samples and has few requirements on their distribution. Afterwards, BikeMate sets the legal riding direction of the bike-way by averaging the directions of bike-way candidates by Eq. 14.

5 EVALUATION

This section presents the evaluation methodologies and detailed performance of BikeMate.

5.1 Evaluation Setup

BikeMate is implemented as a daemon process that runs as a backend service on Android smartphones. Each volunteer is dispatched a smartphone installed with BikeMate and a battery recording logger [30]. Each participant puts the smartphone in his/her trouser pocket during bike riding. BikeMate is launched at the beginning of the riding trip, and it invokes the corresponding sensors to collect measurements. The volunteers are required to perform the different riding behaviors (*i.e.*, lane weaving, standing pedalling, wrong-way riding and normal riding) during each trip and manually label the ground-truth as well as the location information (*i.e.*, the true GPS and the legal riding directions of bike-ways). Each riding trip lasts around 50 minutes. In total 12 volunteers participate in the experiments and 54 bike-ways are covered. We divide the traces into 2640 segments. 80% segments are used for training and the remaining 20% for testing. Fig. 7 illustrates the experiment scenarios of a volunteer. We mainly evaluate BikeMate in terms of accuracy and system overhead.

5.2 Accuracy

Highly accurate detection of dangerous riding behaviors is important to ensure safety and avoid potential accidents. We first evaluate

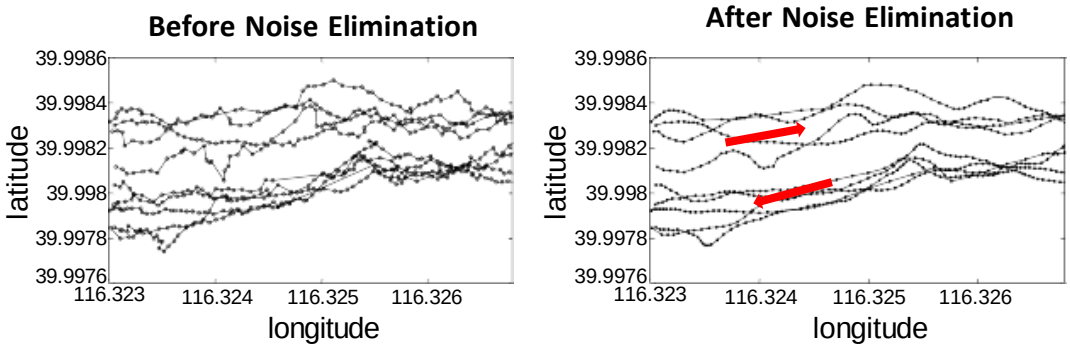

Figure 5: GPS traces of two-side bike lanes before and after pre-processing. There are two notable clusters after pre-processing. The red arrows indicate the inferred directions of two-side bike lanes from GPS sequences sampled by crowdsourcing.

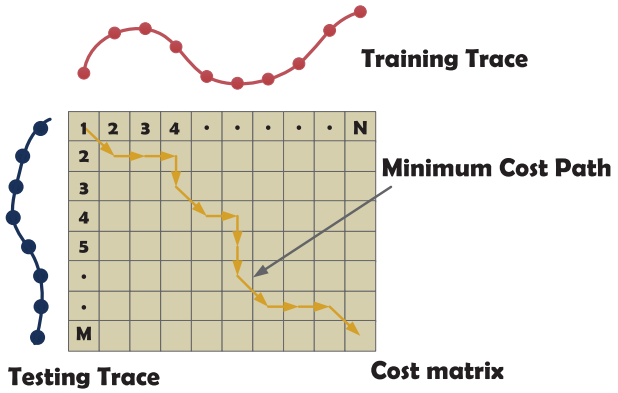

Figure 6: An illustration of DTW based matching.

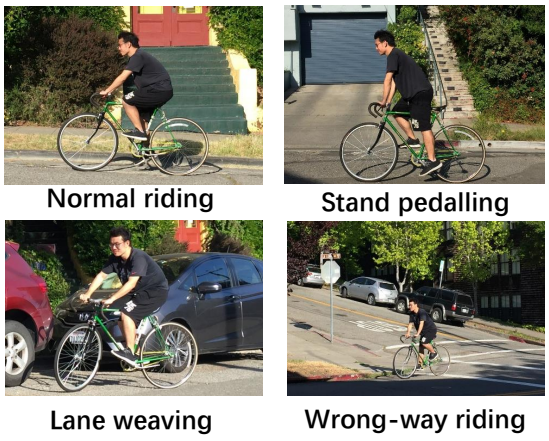

Figure 7: The experimental scenario, where a volunteer performs different riding behaviors.

the accuracy to detect lane weaving and standing pedalling from inertial measurements, and then evaluate the accuracy to detect wrong-way riding from crowdsourced GPS traces.

5.2.1 Performance of Detecting Lane Weaving and Standing Pedalling. Table 1 shows the confusion matrix of identifying normal riding, standing pedalling and lane weaving. Each column represents the instances in an inferred class, while each row represents the instances in an actual class. The results are obtained by training and testing for each participant and average across the 12 users. As shown, the classification accuracy of all the three riding behaviors is higher than 80%. Specifically, the accuracy peaks 88.4% for lane weaving, which is slightly higher than in standing pedalling (87.8%) and normal riding (84.1%). The overall accuracy is 86.8%, demonstrating the remarkable performance of BikeMate on lane weaving and standing pedalling detection.

Table 1: Confusion matrix for lane weaving, standing pedalling and normal riding detection.

Ground Truth	Inference		
	Normal	Stand	Lane Weaving
Normal	84.1%	7.9%	6.9%
Stand	5.6%	87.8%	4.7%
Lane Weaving	10.3%	4.3%	88.4%

5.2.2 Performance of Transfer Kernel Learning. In this experiment, we evaluate the effectiveness of the transfer kernel learning. We first train a riding behavior classifier using measurements from the participants in the training set, and then evaluate its performance using measurements from the rest of the participants in the testing set. Then we apply the transfer kernel learning method on the classifier and test its performance again on the measurements from the rest of the participants. The results are averaged across the 12 participants.

Fig. 8 shows the classification accuracy before and after transfer kernel learning. We observe at least 17% gain in classification accuracy with transfer learning when testing on measurements of new users. The most notable gain is seen for lane weaving (25.6%). This may be because lane weaving has the most distinctive patterns among the three riding behaviors. Therefore the transfer kernel learning scheme can capture such inherent patterns easily and share them among measurements from different users.

Table 2: The configurations of experimental smartphones

Brand	CPU	RAM	ROM	Battery Capacity	Operation System
HUAWEI 4C	8-cores 1.2 GHZ	2 GB	8 GB	3100m Ah	Android 4.4
Galaxy S6	8-cores 2.1 GHZ	3 GB	32 GB	2550m Ah	Android 5.0
HTC Desire A6	8-cores 1.7 GHZ	2 GB	16 GB	2600m Ah	Android 5.0

Table 3: Time Cost of Module and System Delay

Experimental Phones	Feature Extraction		Transfer Kernel Learning (TKL)	Support Vector Machine (SVM)	Total Time Cost
	Acceleration	Angular Acceleration			
HUAWEI 4C	0.11s	0.12s	0.21s	0.17s	0.5s
Galaxy S6	0.08s	0.10s	0.18s	0.13s	0.41s
HTC Desire A6	0.15s	0.13s	0.24s	0.20s	0.59s

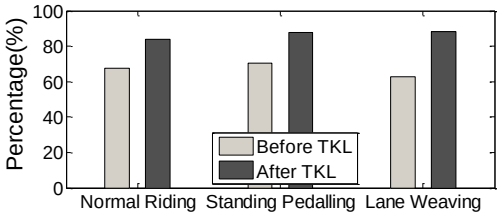

Figure 8: Effectiveness of transfer kernel learning.

5.2.3 Performance of Bike-way Database Construction. An accurate bike-way database constructed from unlabelled GPS traces is an essential prerequisite for wrong-way riding detection. To assess the performance of the bike-way database construction, we compare the GPS traces of bike lanes generated by BikeMate with those manually collected by the participants using the Hausdorff distance. We also compare the riding directions calculated by BikeMate with the true legal riding directions.

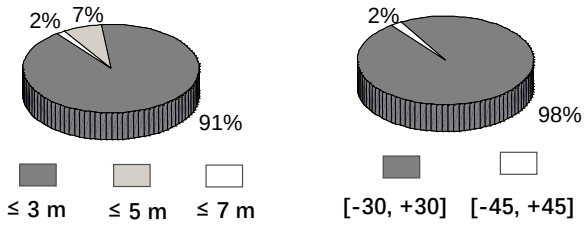

Figure 9: Performance of bike-way database construction: Hausdorff distance between the crowdsourced GPS traces and the true GPS trajectories (left) and the errors of the inferred riding direction compared with the true legal riding directions (right).

The left of Fig. 9 illustrates the Hausdorff distance between the labelled GPS traces and the crowdsourced GPS ones. As shown, more than 91% pairwise distances are within 3m, around 7% distances are less 5m, and only 2% distances are 7m. The right of Fig. 9 shows the divergence of the legal riding direction and the inferred riding direction. As shown, around 98% divergence of directions are within $[-30°, +30°]$, and the other 2% are in $[-45°, +45°]$. Since the wrong-way riding direction can bear a divergence range of $[-90°, +90°]$, the inferred directions generated by BikeMate suffice to assist in wrong-way riding detection.

5.2.4 Performance of Wrong-way Riding Detection. Table 4 shows the accuracy of wrong-way riding detection. We compare the riding direction inferred by BikeMate with the legal cycling direction of the trajectories stored in the database. As shown, the true positive rate is higher than 90%, indicating that BikeMate can detect most wrong-way riding instances. The 13.3% false positive rate shows that normal riding is rarely mistaken as wrong-way riding.

Table 4: The performance of wrong-way riding detection.

Condition	Test	
	Positive	Negative
True	93.2%	6.8%
False	13.3%	86.7%

5.3 System Overhead

As a smartphone-based application, BikeMate needs to incur moderate overhead to the power and computation constrained smartphones. We evaluate the system overhead of BikeMate in terms of delay, CPU utilization and power consumption. To account for the device diversity, we evaluate BikeMate on three different smartphones. Table 2 summarizes the configurations of the smartphones used for system overhead evaluation.

5.3.1 Delay. Since the inference of wrong-way riding is performed at the server end, the main delay of BikeMate comes from the inference for lane weaving, standing pedalling and normal riding. That is, the time spent for feature extraction from inertial sensors, the transfer kernel learning and the SVM classification. We launch a time logger to record the duration of each step.

Table 3 shows the average delays. The time cost of feature extraction fluctuates around 0.1s, and the inference delays of transfer kernel learning (TKL) and Support Vector Machine (SVM) vary around 0.2s and 0.17s, respectively. As BikeMate invokes multithreads to process sensory data in parallel, the total time is the sum

of the maximum processing time among the sensors for feature extraction, the time cost of transfer kernel learning model and the SVM inference. Therefore, the total system delay adds up to around 0.5s, which means BikeMate is able to detect the dangerous riding behaviors within 1s.

5.3.2 CPU Utilization. To measure the CPU usage of BikeMate, we install an application [10] on each smartphone to monitor the CPU occupation while BikeMate is running. We compare the CPU utilization of BikeMate with that of a phone call.

Fig. 10 illustrates the results. The CPU utilization of all the three phones tested are kept relatively in a low and stable level, which are similar to that of a typical phone call. Although BikeMate invokes three kinds of smartphone sensors, the accelerometer and the gyroscope are light-weight, and the GPS runs as a back end service, which will not occupy the CPU. The simplicity of the SVM also ensures low computation overhead.

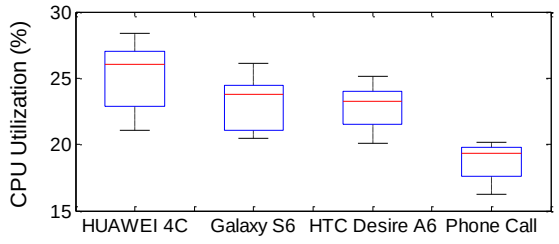

Figure 10: Performance of CPU utilization.

5.3.3 Energy Consumption. As BikeMate is expected to operate continuously during bike riding, it is important that it does not drain the phone battery before the end of the bike trip. We use a battery logger [30] to record the remaining battery level when running BikeMate continuously on the phone.

Fig. 11 shows the results. BikeMate consumes around 3% energy per 10 minutes. Even though BikeMate invokes GPS to track the riding trajectories of users, it only runs at the back end and does not conduct navigation, which can reduce the power consumption.

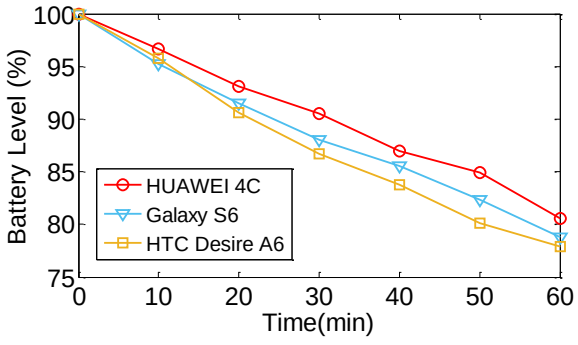

Figure 11: Performance of power consumption.

6 CONCLUSIONS

Preventive bicyclist protection is crucial to promote sustainable transportation such as bicycles. In this paper, we propose BikeMate, a smartphone based dangerous riding behavior monitoring system. It utilizes the embedded inertial sensors and the GPS of smartphones to identify three high-risk behaviors including lane weaving, standing pedalling and wrong-way riding. To improve the usability of the system, BikeMate applies a transfer learning method to enforcing feature sharing to improve the accuracy despite of user-specific differences. BikeMate also leverages crowdsourcing to derive the legal riding directions without prior knowledge. Evaluations with 12 participants validate the effectiveness of BikeMate.

In the future, we plan to include more riding behaviors and optimize the energy consumption of BikeMate. In addition, we consider to extend the capability of commodity WIFI device to tracking the user's riding behavior [33] and even group behiviors [17] of riders. Finally, we will investigate to utilize BikeMate on more types of bike-ways such as one-way bike-ways and single-lane bike-ways.

REFERENCES

[1] Roland Berger. 2016. Bike Sharing 4.0. (June 2016). Retrieved June 19, 2017 from https://goo.gl/EMBDZt.
[2] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proc. KDD*. AAAI Press, Palo Alto, CA, USA, 359–370.
[3] Dongyao Chen, Kyong-Tak Cho, Sihui Han, Zhizhuo Jin, and Kang G Shin. 2015. Invisible sensing of vehicle steering with smartphones. In *Proc. MobiSys*. ACM, New York, NY, USA, 1–13.
[4] Zhongyang Chen, Jiadi Yu, Yanmin Zhu, Yingying Chen, and Minglu Li. 2015. D 3: Abnormal driving behaviors detection and identification using smartphone sensors. In *Proc. SECON*. IEEE, Piscataway, NJ, USA, 524–532.
[5] Hon Chu, Vijay Raman, Jeffrey Shen, Aman Kansal, Victor Bahl, and Romit Roy Choudhury. 2014. I am a smartphone and I know my user is driving. In *Proc. COMSNETS*. IEEE, Piscataway, NJ, USA, 1–8.
[6] Nowy Condro, Meng-Han Li, and Ray-I Chang. 2012. MotoSafe: Active Safe System for Digital Forensics ofMotorcycle Rider with Android. *International Journal of Information and Electronics Engineering* 2, 4 (2012), 612.
[7] Alexandru Dancu, Zlatko Franjcic, and Morten Fjeld. 2014. Smart Flashlight: Map Navigation Using a Bike-mounted Projector. In *Proc. CHI*. ACM, New York, NY, USA, 3627–3630.
[8] M-P Dubuisson and Anil K Jain. 1994. A modified Hausdorff distance for object matching. In *Proc. ICPR*. IEEE, Piscataway, NJ, USA, 566–568.
[9] Evan Gant and Alex Tee. 2010. Light Lane Bicycle. (2010).
[10] GLGJing. 2017. CPU Monitor. (March 2017). Weblink: https://play.google.com/store/apps/details?id=com.glgjing.stark.
[11] John C Gower and GJS Ross. 1969. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics* 18, 1 (1969), 54–64.
[12] Weixi Gu. 2017. Non-intrusive blood glucose monitor by multi-task deep learning: PhD forum abstract. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 249–250.
[13] Weixi Gu, Ming Jin, Zimu Zhou, Costas J Spanos, and Lin Zhang. 2016. MetroEye: Smart Tracking Your Metro Trips Underground.. In *MobiQuitous*. 84–93.
[14] Weixi Gu, Yunxin Liu, Yuxun Zhou, Zimu Zhou, Costas J Spanos, and Lin Zhang. 2017. BikeSafe: bicycle behavior monitoring via smartphones. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, 45–48.
[15] Weixi Gu, Longfei Shangguan, Zheng Yang, and Yunhao Liu. 2016. Sleep hunter: Towards fine grained sleep stage tracking with smartphones. *IEEE Transactions on Mobile Computing* 15, 6 (2016), 1514–1527.
[16] Weixi Gu, Kai Zhang, Zimu Zhou, Ming Jin, Yuxun Zhou, Xi Liu, Costas J Spanos, Zuo-Jun Max Shen, Wei-Hua Lin, and Lin Zhang. 2017. Measuring fine-grained metro interchange time via smartphones. *Transportation Research Part C: Emerging Technologies* 81 (2017), 153–171.
[17] Miao He, Weixi Gu, and Ying Kong. 2017. Group recommendation: by mining users' check-in behaviors. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, 65–68.

[18] J Stuart Hunter. 1986. The exponentially weighted moving average. *Journal of Quality Technology* 18, 4 (1986), 203–210.

[19] Derick A Johnson and Mohan M Trivedi. 2011. Driving style recognition using a smartphone as a sensor platform. In *Proc. ITSC*. IEEE, Piscataway, NJ, USA, 1609–1615.

[20] Balázs Kégl, Adam Krzyzak, Tamás Linder, and Kenneth Zeger. 1999. A polygonal line algorithm for constructing principal curves. In *Proc. NIPS*. MIT Press, Cambridge, MA, USA, 501–507.

[21] Nils Kräuter, Stefan Lösing, Gernot Bauer, Lisa Schwering, and Matthias Seuter. 2016. Supporting safety in cycling groups using LED-augmented gestures. In *Proc. UbiComp Adjunct*. ACM, New York, NY, USA, 889–892.

[22] Mu Li, James Tin-Yau Kwok, and Baoliang Lü. 2010. Making large-scale Nyström approximation possible. In *Proc. ICML*. ACM, New York, NY, USA, 631.

[23] Mingsheng Long, Jianmin Wang, Jiaguang Sun, and S Yu Philip. 2015. Domain invariant transfer kernel learning. *IEEE Transactions on Knowledge and Data Engineering* 27, 6 (2015), 1519–1532.

[24] Ha Quang Minh, Partha Niyogi, and Yuan Yao. 2006. MercerâĂŹs theorem, feature maps, and smoothing. In *Proc. International Conference on Computational Learning Theory*. Springer, Berlin, Germany, 154–168.

[25] U.S. Dept. of Transportation National Highway Traffic Safety Association (NHTSA). 2014. Bicyclists and Other Cyclists. (2014). retrieved June 20, 2017 from https://goo.gl/Zp9wj7.

[26] John C Platt et al. 1999. Using analytic QP and sparseness to speed training of support vector machines. In *Proc. NIPS*. MIT Press, Cambridge, MA, USA, 557–563.

[27] Tarek Sayed, Mohamed H Zaki, and Jarvis Autey. 2013. Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis. *Safety science* 59 (2013), 163–172.

[28] Stephen Smaldone, Chetan Tonde, Vancheswaran K Ananthanarayanan, Ahmed Elgammal, and Liviu Iftode. 2010. *Improving Bicycle Safety through Automated Real-Time Vehicle Detection.* Technical Report DCS-TR-665. Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854.

[29] Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural Processing Letters* 9, 3 (1999), 293–300.

[30] Hwang Ti. 2013. Battery Log. (December 2013). WeblinkijŽ https://play.google.com/store/apps/details?id=kr.hwangti.batterylog&hl=en.

[31] Zheng Yang, Longfei Shangguan, Weixi Gu, Zimu Zhou, Chenshu Wu, and Yunhao Liu. 2014. Sherlock: Micro-environment sensing for smartphones. *IEEE Transactions on Parallel and Distributed Systems* 25, 12 (2014), 3295–3305.

[32] Chuang-Wen You, Nicholas D Lane, Fanglin Chen, Rui Wang, Zhenyu Chen, Thomas J Bao, Martha Montes-de Oca, Yuting Cheng, Mu Lin, Lorenzo Torresani, et al. 2013. Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *Proc. SenSys*. ACM, New York, NY, USA, 13–26.

[33] Han Zou, Yuxun Zhou, Jianfei Yang, Weixi Gu, Lihua Xie, and Costas Spanos. 2017. FreeDetector: Device-Free Occupancy Detection with Commodity WiFi. In *Sensing, Communication and Networking (SECON Workshops), 2017 IEEE International Conference on*. IEEE, 1–5.