Journal of Advanced Transportation

Emerging Information and Communication Technologies for Traffic Estimation and Control

Lead Guest Editor: Anastasios Kouvelas
Guest Editors: Andy Chow, Eric Gonzales, Mehmet Yildirimoglu, and Rodrigo Carlson

Emerging Information and Communication Technologies for Traffic Estimation and Control

Journal of Advanced Transportation

Emerging Information and Communication Technologies for Traffic Estimation and Control

Lead Guest Editor: Anastasios Kouvelas
Guest Editors: Andy Chow, Eric Gonzales, Mehmet Yildirimoglu, and Rodrigo Carlson

Editorial Board

Contents

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 8498054, 2 pages
https://doi.org/10.1155/2018/8498054

Editorial

Emerging Information and Communication Technologies for Traffic Estimation and Control

Anastasios Kouvelas,[1] **Andy Chow**,[2] **Eric Gonzales**,[3] **Mehmet Yildirimoglu**,[4] **and Rodrigo Castelan Carlson**[5]

[1] *École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*
[2] *City University of Hong Kong, Kowloon Tong, Hong Kong*
[3] *University of Massachusetts Amherst, Amherst, MA, USA*
[4] *University of Queensland, Brisbane, QLD, Australia*
[5] *Federal University of Santa Catarina, Florianópolis, SC, Brazil*

Correspondence should be addressed to Anastasios Kouvelas; tasos.kouvelas@epfl.ch

Received 25 March 2018; Accepted 26 March 2018; Published 1 August 2018

Information and communication technologies (ICT) are fundamental components of any development in the field of traffic estimation and control. The design and deployment of multimodal advanced transportation systems involve interdisciplinary expertise and effective utilization of new technologies. Nowadays, there are an increasing number of instrumented networks and transportation deployments around the world, and, as a consequence, the research community in this domain has entered the era of big data. The wealthy amount of traffic data that is available in real time can support the development of modern systematic approaches and methodological tools for estimation and control.

Recent advances in transportation (e.g., emerging vehicle communication and automation technologies) have generated new types of measurements, vehicles and infrastructure communications, and control actuation capabilities. All these enable the transportation community to adapt its research topics accordingly and exploit the opportunities resulting from these new developments to improve traffic estimation and control. There is an essential and timely need for new theoretical techniques and algorithms that can accompany the technological achievements and consequently enhance the performance of transportation systems.

The purpose of this special issue is to publish high quality research papers addressing innovative approaches to improve traffic estimation and control by utilizing recent advances and developments in the area of ICT. Among all the submitted manuscripts, six papers were selected for publication on this special issue after a peer review process, as follows.

V. Lücken et al. propose a density-based statistical clustering approach. They present an integrated approach that combines smart street lamps with traffic sensing technology. Infrastructure-based ultrasonic sensors, which are deployed together with a street light system platform, are utilized for multilane traffic participant detection and classification. They develop an algorithmic approach that combines statistical standardization with clustering techniques from the field of unsupervised learning.

H. Chang and B. Yoon proposed a data-driven k-nearest neighbour nonparametric regression (KNN-NPR) framework that generates short-term traffic volume predictions. The basic characteristic of their approach is high-speed computational performance when dealing with enormous amounts of historical data. To prove the efficacy of their approach, they have conducted an experimental test with large-size traffic volume data.

Y. Liu and Y. Wang presented and compared the vertical and horizontal queueing modelling paradigms for designing and analysing traffic signal timings in oversaturated conditions with limited mobile sensing data. The analytical results were validated by using AIMSUN model. Their paper

addressed the temporal consistency between vertical and horizontal queueing models.

R. A. Paielli proposes a standard Trajectory Specification Language (TSL) based on the Extensible Markup Language (XML) for serializing and communicating aircraft trajectories. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The language is human-readable and may serve as starting point for the development of a communication standard for the Trajectory Specification concept.

X. Hu et al. utilized GPS trajectories from Shenzhen, China, to explore taxi drivers' operation behavior and passengers' demand. Their work focuses on exploring the taxi drivers' operation behavior by utilizing the measurements of activity space and the connection between different activity spaces for different time durations. This research is based on real data and can be helpful for taxi drivers to search for a new passenger and passengers to help them find a taxi's location.

G. A. Klunder et al. presented a study which gives insight into the size of improvement that is possible with individual in-car routing advice based on the actual traffic situation derived from Floating Car Data (FCD). The study uses real loop detector data from the region of Amsterdam, a route generating algorithm for in-car routing advice, and emulated FCD to generate the routing advice. The case with in-car routing advice is compared to the base case, where drivers base their routing decisions on average knowledge of travel times in the network.

Anastasios Kouvelas
Andy Chow
Eric Gonzales
Mehmet Yildirimoglu
Rodrigo Castelan Carlson

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 6986198, 19 pages
https://doi.org/10.1155/2018/6986198

Research Article

Vertical and Horizontal Queue Models for Oversaturated Signal Intersections with Quasi-Real-Time Reconstruction of Deterministic and Shockwave Queueing Profiles Using Limited Mobile Sensing Data

Yongyang Liu,[1] Jingqiu Guo ,[2] and Yibing Wang [1]

[1]*Institute of Intelligent Transportation Systems, Zhejiang University, 866 Yuhangtang Road, Hangzhou 310058, China*
[2]*College of Transportation Engineering, Key Laboratory of Road and Traffic Engineering of the Ministry of Education, Tongji University, 201804, China*

Correspondence should be addressed to Yibing Wang; wangyibing@zju.edu.cn

Received 16 August 2017; Revised 3 January 2018; Accepted 18 January 2018; Published 10 June 2018

Academic Editor: Andy Chow

Deterministic/point/vertical and shockwave/physical/horizontal queueing models are widely used in traffic operation to estimate vehicular queue length and delays at bottlenecks such as signalized intersections. The consistency between the two types of queueing model in terms of their estimation performance has been a subject of debate for decades. This paper reexamines the issue, typically with respect to oversaturated signal intersections, and demonstrates the consistency based on analytical studies and microscopic simulations. While fixed-location sensor data was dominating, it was hardly possible to construct the deterministic or shockwave queueing profile using real data. For this reason, either profile had significance only at a conceptual level and could not be put into practical usage. With the quick spread of mobile sensing data, however, the situation has drastically changed. In this context, this paper also intends to develop an efficient approach to the reconstruction of the deterministic and shockwave queueing profiles in a quasi-real-time manner using very limited mobile sensing data. Microscopic simulations with AIMSUN have demonstrated the efficiency of the approach as well as the analytical results obtained in this paper.

1. Introduction

This paper studies the consistency of the vertical and horizontal queue models as well as the quasi-real-time reconstruction of the deterministic and shockwave queueing profiles using very limited mobile sensing data for oversaturated signal intersections.

The vertical and horizontal queue models are two distinctive queueing models widely used in transport and traffic engineering. The vertical queue model provides a physically impossible but conceptually valid description of queueing processes. It postulates that any vehicle that has to queue before passing a bottleneck is stacked in a vertical pile at the bottleneck, on top of any other such vehicles that arrived earlier. When the queue is released, departing vehicles exit from the bottom of the queue. A vertical queue does not occupy any road space and has no influence on upstream approaching vehicles. It is therefore also called a point queue. In contrast, the horizontal queue model formulates the formation and dissipation of a physical queue. Figure 1 illustrates the two types of queue with respect to a signalized intersection.

The deterministic queueing profile is commonly used to describe the vertical queueing process. It is based on cumulative diagrams, which were initially introduced to transport by Moskowitz and Newman [1] and Gazis and Potts [2] and later highlighted by Newell for the potential as a traffic analysis tool (Daganzo [3]). On the other hand, the shockwave queueing profile is based on traffic flow theory [4, 5] to describe the spatiotemporal dynamics of the horizontal queues. Both queueing profiles are depicted in Figure 2 for an oversaturated signal intersection.

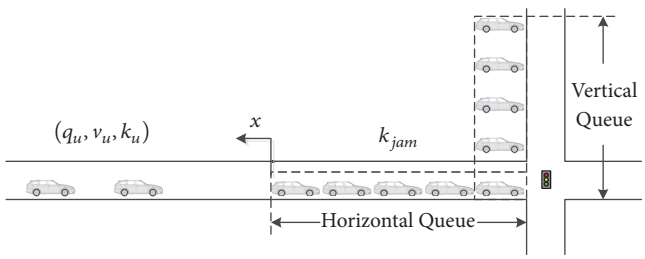


Figure 1: The horizontal and vertical queue models for a signalized intersection (x and k_{jam} are given in Figure 5).

The consistency of the vertical and horizontal queue models or equivalently of the deterministic and shockwave queueing profiles (in terms of their modeling performance) is of much interest and has been a subject of debate for decades. Several textbooks and handbooks have described the two models or two queueing analysis methods as separate tools for treating bottleneck problems (see, e.g., McShane and Roess [6]). Makigami et al. [7] studied the two analysis methods using a three-dimensional (space-time-queue) model and demonstrated their consistency. Wirasinghe [8] proved that the two methods delivered identical results in the total travel delay. Michalopoulos and Pisharody [9] however commented that the deterministic profile overestimated travel delays, while Daganzo [10] pointed out a flaw in that work and emphasized that, with the flaw removed, the two methods were still consistent. More than a decade later, the inconsistency issue was again raised by Chin [11], McShane and Roess [6], and Nam and Drew [12], whereas Lawson et al. [13], Erera et al. [14], Daganzo [3], and Lovell and Windover [15] underlined the consistency once more. More recently, Rakha and Zhang [16] and Yi et al. [17] concurred with the consistency but tried to improve Nam and Drew's approach [12].

The vertical and horizontal queue models are quite different in their operational mechanisms, which is very much the reason for the consistency issue to be raised from time to time. With the literature review, we see that thorough and analytical investigations on the issue still lack. This paper intends to close the debates via a systematic study, particularly for the case of oversaturated signal intersections. This is actually the first motivation of the current work.

Traffic signals are one of the crucial components of traffic networks. Queue length [8, 18–20] and total delay [6, 8, 20, 21] of vehicles due to signal operation are among the most important performance measures for traffic signal control. Many relevant studies were conducted using the deterministic queueing profile [8, 10, 12, 15, 20] or shockwave queueing profile [8, 18, 19] with fixed-location sensing data (e.g., loop detector data). As a matter of fact, both deterministic and shockwave queueing profiles are individual-vehicle-trajectory oriented. Fixed-location sensors can hardly provide sufficient information for constructing either profile for the purpose of traffic signal control. For this reason, while fixed-location sensing technologies were dominating, both queueing profiles only had significance at a conceptual level and could not offer strong operability in practice. In recent years, however, the emergence of mobile sensing technologies based on connected vehicles (CVs) has been attracting increasingly more attention. Equipped with GPS (or other tracking devices) and wireless communication systems, CVs are able to provide real-time information (of their positions, speeds, and acceleration), thus leading in total to better spatial coverage of traffic conditions. This has also offered opportunities of reexamining the two queueing models/profiles for traffic modeling and signal control.

A number of studies have been conducted using mobile sensing data, to address shockwave profile reconstruction for the purpose of queue length estimation [22–26]. The focus of those works is on the identification of some critical points on vehicle trajectories based on the shockwave model. For instance, [22] tried to figure out the critical points that fall on the congestion forming shockwave curves (CFSC) to determine vehicle trajectories. The work in [24] fitted vehicle trajectories with piecewise linear lines and took critical points as the intersection of the lines. The work in [25] fixed the critical points using both speed and acceleration information. The work in [26] formulated the construction of CFSC as a convex optimization problem. In addition, [27–29] studied queueing delay patterns using mobile-sensing-based travel time data and estimated queue length for signalized intersections. So far, still missing is a simple and efficient approach to the reconstruction of shockwave queueing profiles using mobile sensing data, and this is the second motivation of the current work.

It should be noted that only the shockwave queueing profiles can be directly reconstructed using mobile sensing data. Due to its abstract nature, the deterministic queueing profile can only be built upon the corresponding shockwave queueing profile in consideration of the consistency between the two profiles.

To sum up, the objective of this work is twofold. First, it aims to close the debates on the consistency of the two queueing models via some systematic and analytical studies. Second, it intends to develop an efficient approach to the reconstruction of the shockwave and deterministic queueing profiles in a quasi-real-time manner using very limited mobile sensing data. The work was done with regard to oversaturated signal intersections. Microscopic simulation tool AIMSUN was used to deliver emulated mobile sensing data and evaluate the accuracy of queueing profile reconstruction, with various penetration rates of connected vehicles taken into account.

The remainder of the paper is organized as follows. Section 2 introduces the vertical and horizontal queue models as well as the deterministic and shockwave queueing profiles with respect to an oversaturated signal intersection. The consistency properties of the two queueing models/profiles are presented in Section 3. Section 4 focuses on the quasi-real-time reconstruction of the two queueing profiles using very limited mobile sensing data. Section 5 reports on simulation studies, with satisfactory results delivered. Section 6 concludes the paper.

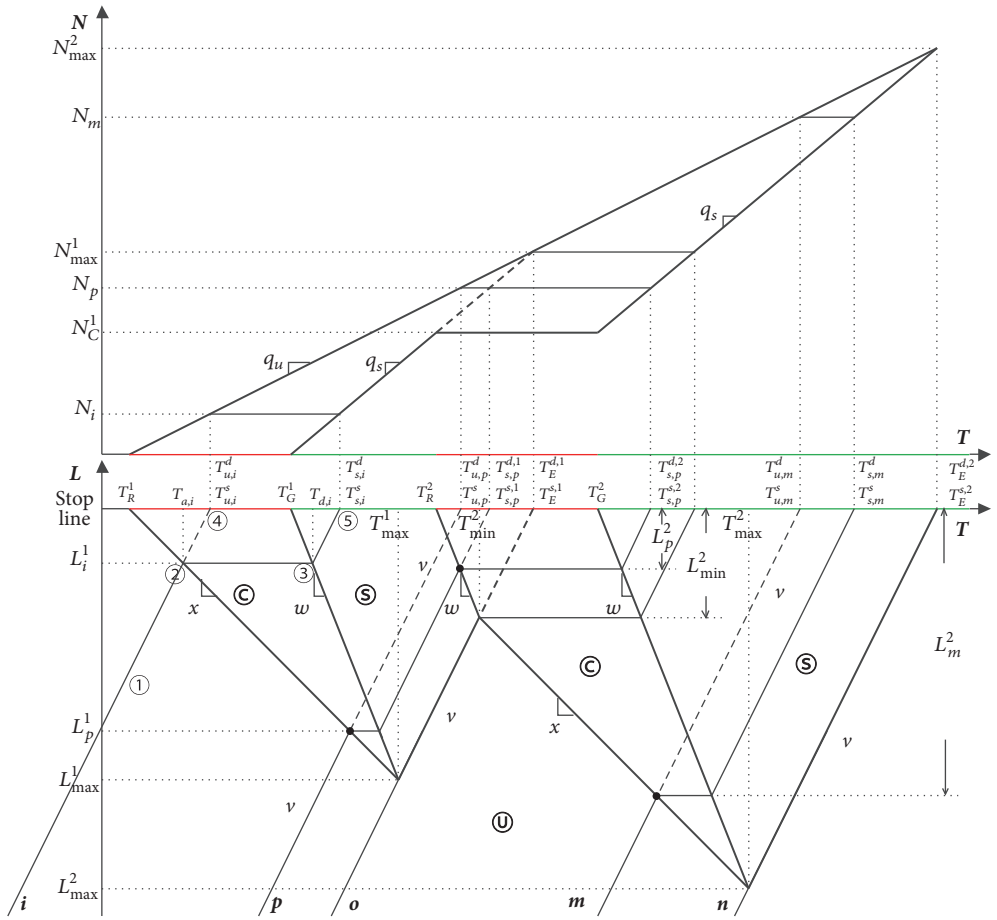


Figure 2: Deterministic and shockwave queueing profiles for an oversaturated signal intersection (traffic conditions Ⓒ, Ⓢ, and Ⓤ are defined with Figure 5).

2. Queueing Models for Oversaturated Signal Intersections

2.1. Vertical and Horizontal Queue Models. With reference to Figure 1, either queueing model takes as an input-output system the displayed road link between the upstream entry and the downstream intersection. Consider one vehicle and its counterpart enter the road link at the same time and experience the horizontal and vertical queueing, separately. The two vehicles should pass the stop line at the intersection at the same time. This is actually a minimum requirement for the two queueing models; i.e., given the same input, the model outputs should be the same, despite quite different internal mechanism of queueing.

The two queueing models can be further examined with Figures 3 and 4. Figure 3 depicts a queue forming process over a red signal phase at the intersection and focuses on three typical time instants: (a) both vertical and horizontal queues exist with three vehicles A, B, and C, while D is approaching either queue; (b) D just joins the horizontal queue while it is still running towards the vertical queue at the stop line; (c) D reaches the stop line and becomes the new tail of the vertical queue. Focusing on vehicle D, we see

(i) it makes no difference to the two types of queue before time instant (b);

(ii) it joins the horizontal queue earlier than the vertical queue;

(iii) after joining the horizontal queue it keeps its original speed until joining the vertical queue;

(iv) the process with which it joins the vertical queue as depicted in Figure 2(c) is completed instantaneously.

Figure 4 displays a queue dissolving process and focuses on three typical time instants: (a) vehicles are queueing right before the end of a red signal phase; (b) vehicles A and B have already passed the stop line, and vehicle C is waiting in the vertical queue at the stop line while it has departed from the horizontal queue and is running towards the stop line; (c) vehicle C is passing the stop line from both queues. The process that C departs from the vertical queue to pass the stop line is completed instantaneously. In summary,

(i) a horizontal queue makes a distinction between "departing from the queue" and "passing the stop line", while a vertical queue does not;

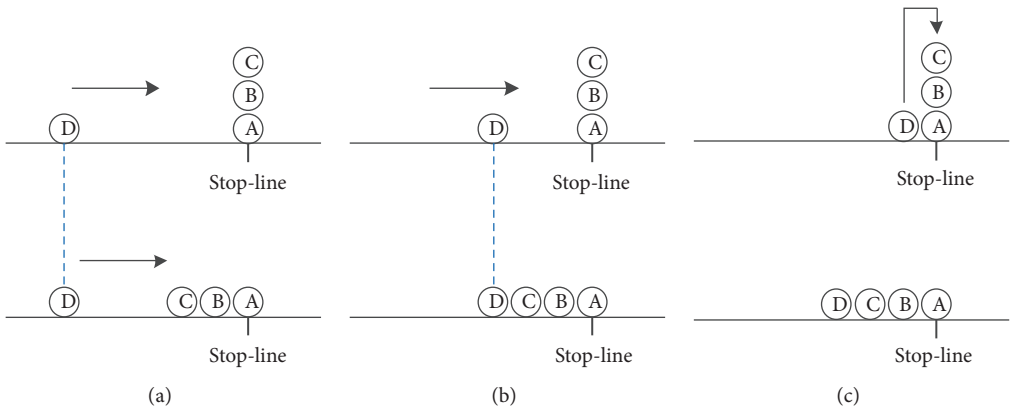


Figure 3: Vehicles arrive to join a vertical queue (upper) and a horizontal queue (lower) over a red signal phase: (a) when D is approaching either queue; (b) when D just joins the horizontal queue; (c) when D just joins the vertical queue.

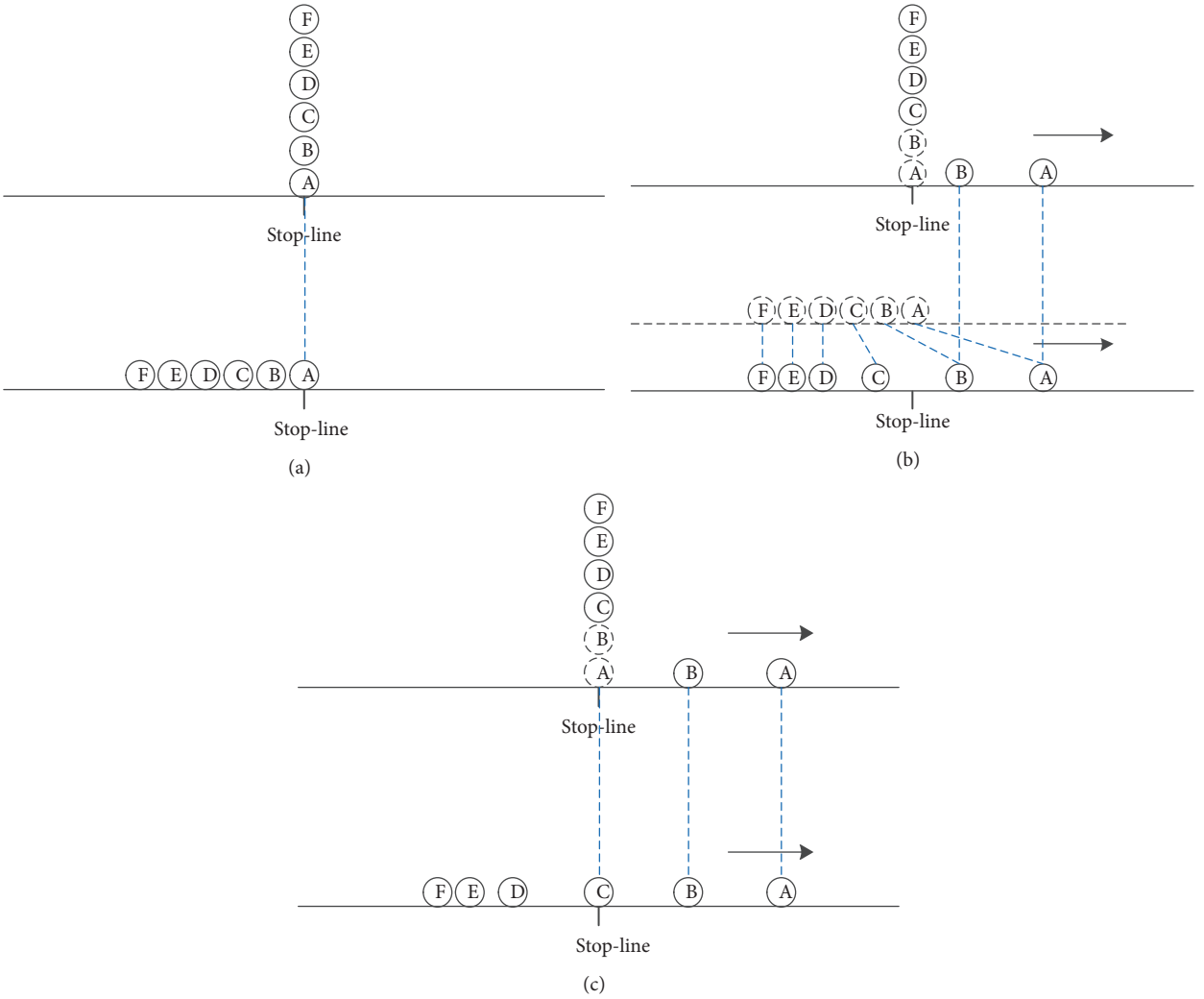


Figure 4: Vehicles depart from a vertical queue (upper) and a horizontal queue (lower): (a) vehicles are all queueing before the end of a red phase; (b) C has departed from the horizontal queue but is still waiting in the vertical queue; (c) C is passing the stop line from both queues.

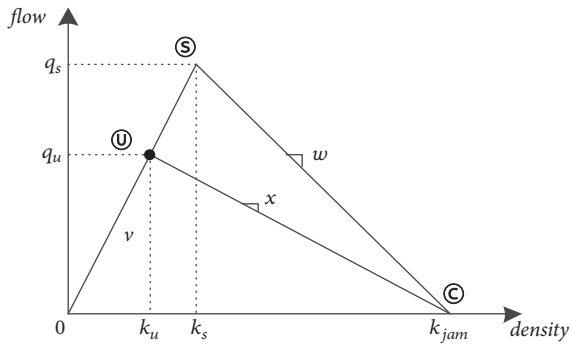


Figure 5: A triangular fundamental diagram.

(ii) a horizontal queue vehicle and its counterpart in the vertical queue pass the stop line at the same time.

2.2. Deterministic and Shockwave Queueing Profiles. With reference to Figure 1, traffic conditions along the road link and at the intersection are defined with the fundamental diagram in Figure 5. More specifically $U(q_u, k_u, v)$, $S(q_s, k_s, v)$, and $C(0, k_{\text{jam}}, 0)$ address the upstream arrival flow, the capacity flow at the intersection, and the queueing condition on the road link, with x and w denoting the queue forming and dissolving shockwaves.

Without loss of generality, Figure 2 plots the deterministic and shockwave queueing profiles for the intersection under signal oversaturation over two cycles. A number of definitions and notations are first given. On the time axes, T_R^1, T_G^1, T_R^2, and T_G^2 represent the start and end times of red phases. The superscripts "d" and "s" of symbols on the time axes refer to the "deterministic" and "shockwave" queueing profiles, respectively. The two profiles are connected via individual vehicle trajectories; see, e.g., vehicles i, p, and m. As a consequence, a number of time instants appear in pairs, e.g., $T_E^{d,1} = T_E^{s,1}$, $T_E^{d,2} = T_E^{s,2}$, $T_{u,\varphi}^d = T_{u,\varphi}^s$, $T_{s,\varphi}^d = T_{s,\varphi}^s$ ($\varphi = i, p, m$). $T_E^{d,2}$ (or $T_E^{s,2}$) denotes the time instant by which vehicles that are delayed over the two cycles have all been served, while $T_E^{d,1}$ (or $T_E^{s,1}$) is a virtual time instant by which all vehicles delayed over the first cycle would have been served without signal oversaturation. $T_{u,\varphi}^d$ or $T_{u,\varphi}^s$ denotes the time vehicle φ joins the vertical queue, and $T_{s,\varphi}^d$ or $T_{s,\varphi}^s$ denotes the time vehicle φ leaves the vertical queue (or equivalently passes the stop line with both queues). The timewise consistency between the two profiles is discussed later with Theorems 1 and 2. Moreover, $T_{\max}^1$ and $T_{\max}^2$, defined only for the shockwave profile, are the time instants the physical queues spill back to reach their far ends.

On the N axis of the deterministic profile, N_C^1 refers to the actual number of vehicles served by the end of the first cycle, $N_{\max}^2$, corresponding to $T_E^{d,2}$ (or $T_E^{s,2}$), refers to the total number of vehicles that are delayed but eventually served over the two cycles, and $N_{\max}^1$, corresponding to $T_E^{d,1}$ (or $T_E^{s,1}$), is a virtual quantity addressing the number of vehicles that would have been served by the end of the first cycle without signal oversaturation. In addition, N_i, N_p, and N_m denote the accumulative numbers for vehicles i, p, and m in the vertical queue.

On the L axis of the shockwave profile, L_φ^θ denotes the distance between vehicle φ and the stop line when it joins the horizontal queue in cycle θ. $L_{\max}^1$ and $L_{\max}^2$, corresponding to $T_{\max}^1$ and $T_{\max}^2$, respectively, address the far ends that the physical queues can reach.

In the case of signal oversaturation over two cycles, some vehicles are involved with the horizontal queue once, while some others may experience queueing twice. With "single" queueing, a vehicle φ joins and leaves the vertical queue at $T_{u,\varphi}^d$ and $T_{s,\varphi}^d$ (e.g., vehicles i and m). With "double" queueing, a vehicle φ (e.g., vehicle p) joins the vertical queue at $T_{u,\varphi}^d$ and would leave it at $T_{s,\varphi}^{d,1}$ if there was no oversaturation but factually leaves it at $T_{s,\varphi}^{d,2}$. Accordingly, the timewise consistency between the two profiles reads $T_{u,\varphi}^d = T_{u,\varphi}^s$; $T_{s,\varphi}^d = T_{s,\varphi}^s$ or $T_{s,\varphi}^{d,1} = T_{s,\varphi}^{s,1}$ and $T_{s,\varphi}^{d,2} = T_{s,\varphi}^{s,2}$.

When a vehicle φ joins a horizontal queue, its distance from the stop line follows a different way of definition. If φ joins the horizontal queue only in cycle 1, then only L_φ^1 is defined (e.g., $\varphi = i$); if it joins the queue only in cycle 2, only L_φ^2 is defined (e.g., $\varphi = m$); if it joins the queue twice, both L_φ^1 and L_φ^2 (e.g., $\varphi = p$) make sense.

3. Consistency Properties of the Queueing Profiles

The consistency properties of the two queueing profiles are described via four theorems and one corollary. The mathematical proof is found in Appendices A and B. To the best of our knowledge, these properties were not explicitly discussed, especially for oversaturated signal intersections. The theorems and corollary are presented on the basis of Figure 2, with respect to signal oversaturation over two cycles, but also apply to more general cases.

Theorem 1. *For a signal cycle θ ($\theta = 1, 2$), $T_E^{s,\theta} = T_E^{d,\theta}$.*

Proof. See Appendix A. □

With reference to Figure 2, $T_E^{s,1}$ and $T_E^{d,1}$ are two virtual quantities while $T_E^{s,2}$ and $T_E^{d,2}$ are well defined. The theorem says that the two types of queue are cleared at the same time in the case of either undersaturation ($\theta = 1$) or oversaturation ($\theta = 2$).

Theorem 1 sets a basis for the proof of the other theorems and corollary. Following Theorem 1, $T_E^{d,\theta}$ and $T_E^{s,\theta}$ ($\theta = 1, 2$) are replaced with T_E^θ ($\theta = 1, 2$) in the sequel.

Theorem 2. *Consider a vehicle φ and its counterpart, which go through the vertical and horizontal queues separately. Then,*

(1) the vehicles do not join the two queues at the same time, but $T_{u,\varphi}^s = T_{u,\varphi}^d$;

(2) the vehicles do not depart from the queues at the same time but pass the stop line at the same time, i.e., $T^{s}_{s,\varphi} = T^{d}_{s,\varphi}$ *or* $T^{s,\theta}_{s,\varphi} = T^{d,\theta}_{s,\varphi}$ $(\theta = 1, 2)$.

Proof. See Appendix A. □

Regarding the property demonstrated with Theorem 2-(2), see also Section 2.1.

Corollary 3. *Given a vehicle* φ*, its total horizontal queue delay* d^{s}_{φ} *is equal to its vertical queue delay* d^{d}_{φ}.

Proof. See Appendix A. □

This may also be checked graphically with, e.g., vehicle i in Figure 2. The trajectory of vehicle i in the shockwave profile is ②③⑤, while ②④⑤ is its vertical queue trajectory in time. Clearly, vehicle i spends the same period of time in both profiles, i.e., between ④ and ⑤ for the vertical queue and between ② and ③ for the horizontal queue.

Theorem 4. *For any vehicle* φ *in a signal cycle* θ $(\theta = 1, 2)$,

$$\begin{aligned} N_{\varphi} - N_{C}^{\theta-1} &= L_{\varphi}^{\theta} \cdot k_{\text{jam}} \\ T_{G}^{\theta} \le T_{s,\varphi}^{s,\theta} &\le T_{E}^{\theta}, \end{aligned} \tag{1}$$

where $N_{C}^{0} = 0$, $N_{C}^{1} = q_{s} \cdot (T_{R}^{2} - T_{G}^{1})$.

Proof. See Appendix A. □

Any vehicle φ and its counterpart do not join the horizontal queue and the corresponding vertical queue at the same time, but their positions in the queues (N_{φ} and L_{φ}^{θ}) are interrelated as stated above. With reference to Figure 2, only L_{i}^{1} is defined for vehicle i; then $N_{i} = L_{i}^{1} \cdot k_{\text{jam}}$; both L_{p}^{1} and L_{p}^{2} are defined for vehicle p; then $N_{p} = L_{p}^{1} \cdot k_{\text{jam}}$ and $N_{p} - N_{C}^{1} = L_{p}^{2} \cdot k_{\text{jam}}$; only L_{m}^{2} is defined for vehicle m; then $N_{m} - N_{C}^{1} = L_{m}^{2} \cdot k_{\text{jam}}$.

It is noted that Theorem 4 also applies to more general cases with $\theta > 2$, where $N_{C}^{\theta} = q_{s} \cdot (T_{R}^{\theta+1} - T_{G}^{\theta})$. The proof is omitted.

Corollary 5. *For any vehicle* φ *that experiences queueing more than once,*

$$L_{\varphi}^{2} = L_{\varphi}^{1} - \frac{\left(T_{R}^{2} - T_{G}^{1}\right) \cdot q_{s}}{k_{jam}} \tag{2}$$

Proof. See Appendix A. □

Given a vehicle φ that experiences queueing more than once over multiple signal cycles, the corollary formulates the relation of this vehicle's queue length in two neighboring cycles.

It is noted that Corollary 5 also applies to more general cases with $\theta > 2$:

$$L_{\varphi}^{\theta+N} = L_{\varphi}^{\theta} - \frac{\sum_{i=1}^{N}\left(T_{R}^{\theta+i} - T_{G}^{\theta+i-1}\right) \cdot q_{s}}{k_{\text{jam}}} \tag{3}$$

The proof is omitted.

Theorem 6. *The total delays created by the vertical and horizontal queueing models are equal, i.e.,* $S_{V} = S_{H} \cdot k_{jam}$*, where* S_{V} *is the total area of the deterministic queueing profile over the whole period* $[T_{R}^{1}, T_{E}^{d,2}]$ *in Figure 2, while* S_{H} *is the total area of the two shockwave polygons over periods* $[T_{R}^{1}, T_{max}^{1}]$ *and* $[T_{R}^{2}, T_{max}^{2}]$ *in Figure 2.*

Proof. See Appendix A. □

So far what has been discussed with Figure 2 is the case of a constant traffic demand. The theorems and corollary can certainly be extended to the case of a varying traffic demand (Figure 6), and the corresponding proof is found in Appendix B.

4. Quasi-Real-Time Reconstruction of Queueing Profiles Using Limited Mobile Sensing Data

Conventional fixed-location detectors such as loops, radars, and cameras cannot provide sufficient or suitable measurements to construct the deterministic and shockwave queueing profiles. Albeit well-known in traffic engineering, the vertical/horizontal queue models and the deterministic/shockwave queueing profiles have long been considered only at a conceptual level. With the emergence of the mobile sensing technologies, however, the increasingly wider spread of mobile sensing data has made it possible to construct both queueing profiles in quasi-real time. In fact, what can be directly created using mobile sensing data is the shockwave queueing profile, but with the consistency properties presented in Section 3, especially Theorems 1, 2, and 4, the deterministic profile can also be constructed. As an illustrative example, Figure 7(a) presents the shockwave profiles constructed using mobile sensing data emulated with the microscopic simulation tool AIMSUN. The shockwave profiles over the five most congested cycles are highlighted, over which the corresponding signal intersection is oversaturated. Figure 7(b) displays the associated deterministic profile. The approaches employed to create both profiles are described in this section, and more simulation details are found in Section 5.

4.1. Basic Ideas. Given an intersection as shown in Figure 8, its arriving flow may consist of several components, according to the signal phase setting at the intersection right upstream: the through movement in phase (a); the right-turn movement during phase (c); the left-turn movement within phase (d). As an illustrative example, Figure 7(a) presents the correspondence of flows reaching the downstream intersection to the upstream signal phases as given in Figure 8.

Based on vehicle to vehicle/infrastructure communication, assume with reference to Figure 8 that the start and end times of each signal phase of the two intersections are known, and each movement towards the downstream intersection as shown in Figure 8 (i.e., each flow in Figure 7(a)) contains at least two connected vehicles (CVs). Then, each linear section of the congestion forming shockwave curve (CFSC) as shown

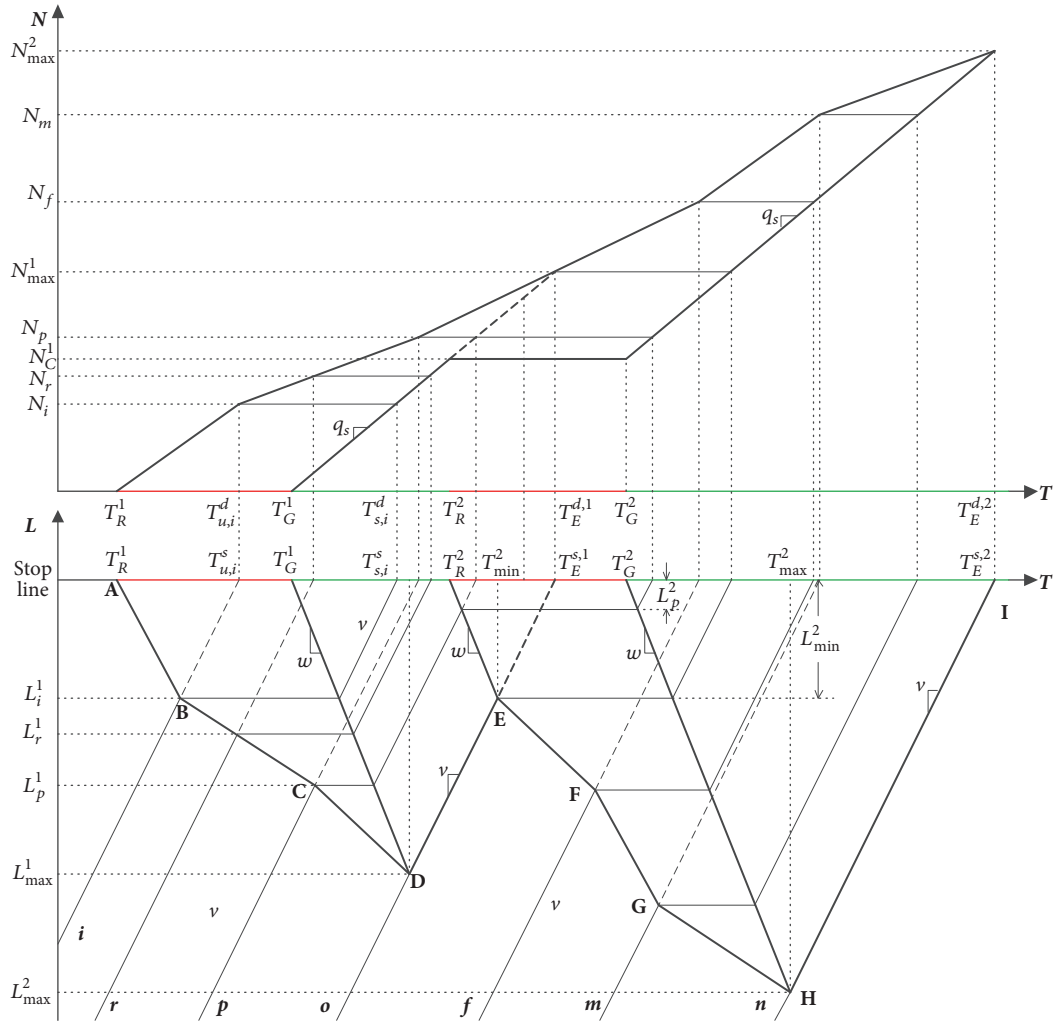


Figure 6: Deterministic and shockwave profiles under varying traffic demands.

in Figure 9 contains at least two measured points (e.g., y_0 and y_1), and the whole CFSC can be basically determined in quasi-real time. Similarly, the congestion dissolving shockwave w can also be determined. w is supposed to be constant by Figure 5, which is also confirmed with Figure 7(a) in simulation. Clearly, the more the CVs are available, the more accurately the shockwave profile is identified. This way of determining the shockwave profile was also considered in [12, 13, 15] via linear optimization. Once the shockwave profile is fixed, the deterministic profile can be readily determined as well.

Consider a CV i in Figure 9. The information of its positions and speeds as well as the corresponding time instants is available along its trajectory, e.g., at points y_1, y_3, and y_4. By Theorem 2 and the definition of the vertical queue, the time it joins the deterministic queueing profile at y_5 is equal to the time instant it reaches virtually y_2. y_2 can be approximately determined as $(1/n) \cdot \sum_{k=1}^{n} [t_k + L(t_k)/v(t_k)]$, where n denotes a given number of data points collected right before it joins the horizontal queue at y_1, $L(t_k)$ denotes its distance from the stop line at each time instant t_k, and $v(t_k)$ denotes the instantaneous speed. By Theorem 2, y_6 and y_4 are consistent in time. Moreover, based on Theorem 4, it is straightforward to get N_i for the deterministic profile with measured L_i in the shockwave profile. Thus, the full coordinates of y_5 and y_6 are fixed. As such, as long as two CVs are available for each portion of the arrival flow, both shockwave and deterministic queueing profiles can basically be determined; see, e.g., Figure 9.

4.2. Creating a Shockwave Profile

4.2.1. The Case of Two Connected Vehicles in Each Subflow. As previously mentioned, traffic arriving at an intersection can be decomposed into several flow portions as regulated by the signal phases at the intersection right upstream. There may also be a number of periods over which no vehicle arrives at the intersection. Such periods may appear in the middle of a CFSC, e.g., between flows ① and ② in Figure 7(a), or at the end of a CFSC, e.g., between flows ② and ③ in

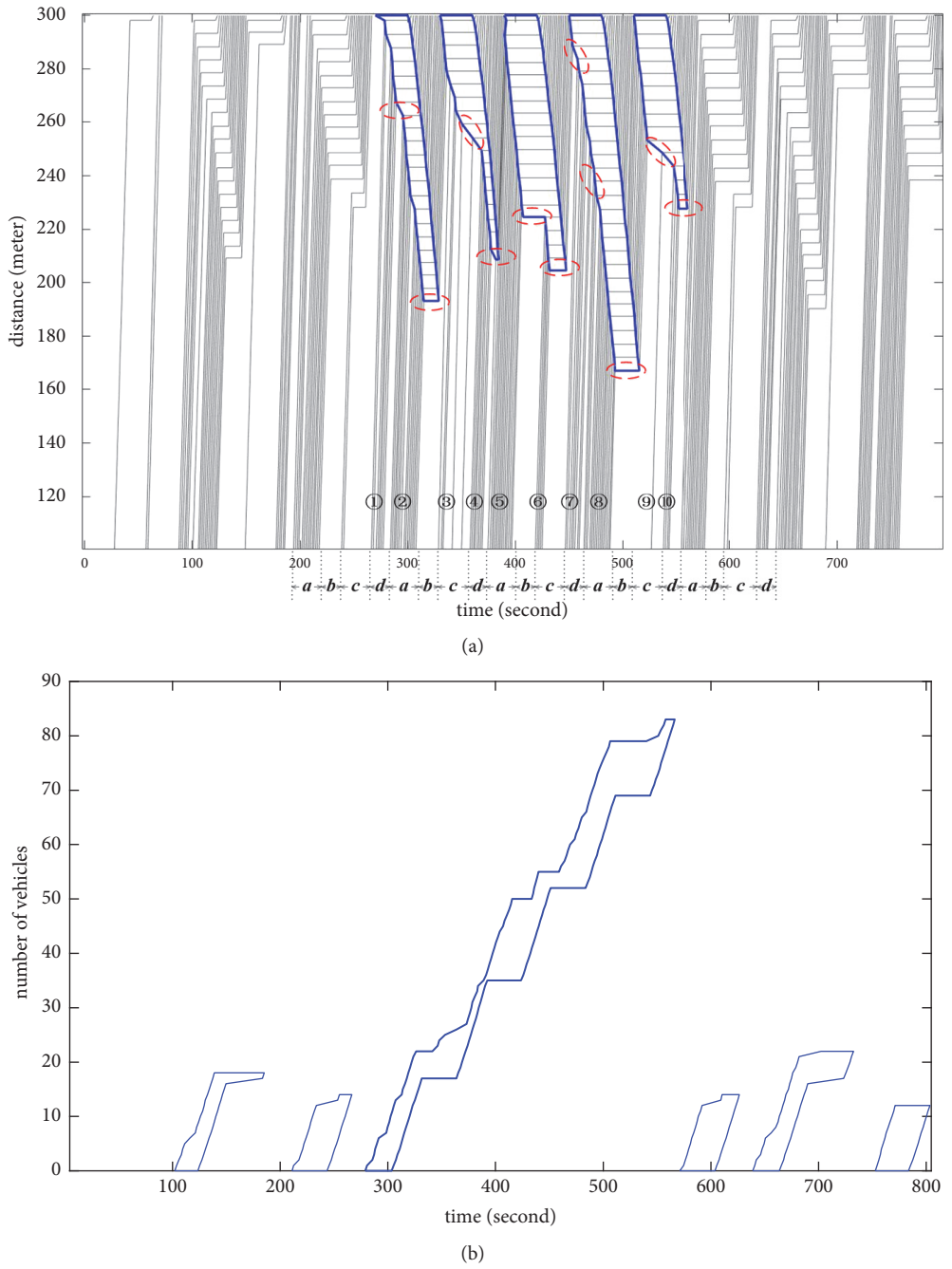


FIGURE 7: Queueing profiles based on microscopic simulation with 100% penetration rate of CVs: (a) shockwave profile (with the signal phases at the upstream intersection); (b) the corresponding deterministic profile.

Figure 7(a). Note that there are two possibilities concerning no vehicle arrival at a subject intersection: (1) the signal phase of the upstream intersection is prohibitive (e.g., phase (b) in Figure 8 and Figure 7(a)); (2) the upstream signal phase is permissive but by coincidence there is no demand; see in Figure 7(a), e.g., phase (d) in relation to flow ① and phase (c) to flow ⑥. Either case may occur in the middle of or at the end of a CFSC. It is stipulated in this work that if no vehicle arrival happens in the middle of a CFSC, the portion of CFSC concerning (1) is plotted as a horizontal line, e.g., between flows ⑤ and ⑥ in Figure 7(a), and the portion concerning (2) is plotted as a line, e.g., between ① and ② in Figure 7(a). However, if either (1) or (2) happens at the end of a CFSC, the corresponding portion is naturally plotted as a horizontal line, e.g., between ② and ③ and between ⑥ and ⑦ in Figure 7(a).

It is noted that Figure 7(a) plots the case of 100% penetration rate of CVs, where every trajectory displayed corresponds to one specific CV. However, the market penetration rate of CVs will be low in the foreseeable future, which means,

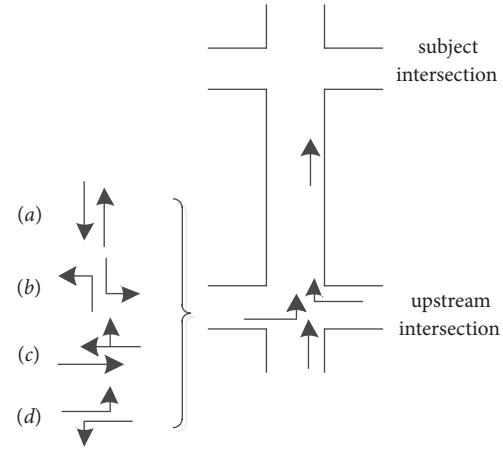


Figure 8: Arrival flow from upstream intersection.

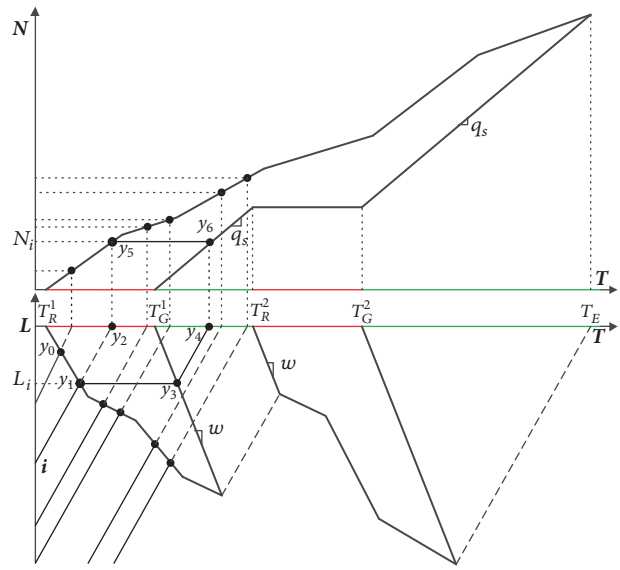


Figure 9: Two queueing profiles with variant upstream arrival flow.

in the context of Figure 7(a), only a very small number of vehicle trajectories displayed are actually observable. It is then a question how to construct the shockwave profile and accordingly the deterministic profile using very limited vehicle trajectory data. To be realistic, we assume in this work that each flow as shown in Figure 7(a) contains at most two CVs.

While constructing the shockwave profile using very limited mobile sensing data, it is common to encounter the situation of missing data. Six possible cases are presented in Figure 10. Figures 10(a) and 10(b) address the occurrence of missing data in the middle of CFSCs, while Figures 10(c)–10(f) address the cases at the end of CFSCs. Note that on top of "low penetration of CVs", "no vehicle arrival" as previously discussed also contributes to the result of missing data.

In Figure 10, CVs r and s correspond to one green phase of the upstream intersection, while CVs e and f correspond to the next green phase. Each of subfigures (b), (d), and (f) involves a prohibitive phase (e.g., phase (b) in Figure 8 and period OD in Figure 10(b)), while none of subfigures (a), (c), and (e) involves such a phase. Moreover, m represents a virtual vehicle that leaves the upstream intersection at the start of the next green phase, and virtual vehicle n leaves at the end time of the current green phase, which is followed by a prohibitive phase. Taking Figure 7(a) as the ground truth (with the 100% market penetration), all cases in Figure 10 except (e) and (f) are encountered while estimating both profiles using very limited mobile sensing data as sampled from Figure 7(a). More details are given in Section 5.

In what follows, we discuss how to treat the cases shown in Figure 10, with various situations of missing data taken into account, so as to reconstruct the shockwave profiles. In Figure 10(a), CVs r and s are from one green phase of the upstream intersection while CVs e and f correspond to another. It is not difficult to distinguish this since the information of all signal phases and of CVs are available. As such, y_2 is related to the last CV of that green phase, and y_3 is to the first CV of the next green phase. Then, two linear portions of CFSC can be determined, respectively, with CV data points y_1, y_2, y_3, and y_4. Note that the start time of the upstream green phase for e and f is known. Let a virtual vehicle m leave the upstream intersection at this time instant; the trajectory of m can be drawn with its slope equal to the mean speed of e and f. As such, points D and C can be fixed. It is noted that the portion of the congestion forming shockwave curve (CFSC) between y_1 and D belongs to the previous green phase. As such, there exist two extreme but possible cases: (1) y_2CD, i.e., section y_1C is with a constant traffic volume while no vehicle arrives over period CD; (2) y_2D, i.e., sections y_1y_2 and y_2D are with different traffic volumes. Moreover, between cases (1) and (2), there are a number of intermediate possibilities as displayed. Note that the ground truth can be any of the above cases. Based on our simulation results in Section 5, we believe case y_2D is more likely to happen than y_2CD.

Figure 10(b) refers to the case that a no-flow phase OD is involved. In this case, let virtual vehicle n leave the upstream intersection at the end time of the green phase associated with vehicles r and s; then the trajectory of n is fixed with its speed equal to the mean speed of vehicles r and s. As such, point O can be determined. The treatment of the portion between y_2 and O in Figure 10(b) is the same as that between y_2 and D in Figure 10(a).

Figures 10(c)–10(f) address the case that the data is missing at the end of a CFSC. First, we consider Figure 10(c), where CVs e and f from the next green phase of the upstream intersection do not join the queue in the cycle of the subject intersection for vehicles r and s, and this is also assumed to be the case for all vehicles from the next green phase (among them only e and f are observable). Let virtual vehicle m leave the upstream intersection at the start of the next green phase, and then points C and D can be fixed with the information of y_1, y_2, y_3, and y_4. Figure 10(d) refers to a similar case but the signal phase next to that for CVs r and s is a prohibitive phase. Let virtual vehicle n leave the upstream intersection at the end of the current green phase, and points C and D can be fixed. Note that CD in Figure 10(c) or 11(d) represents an

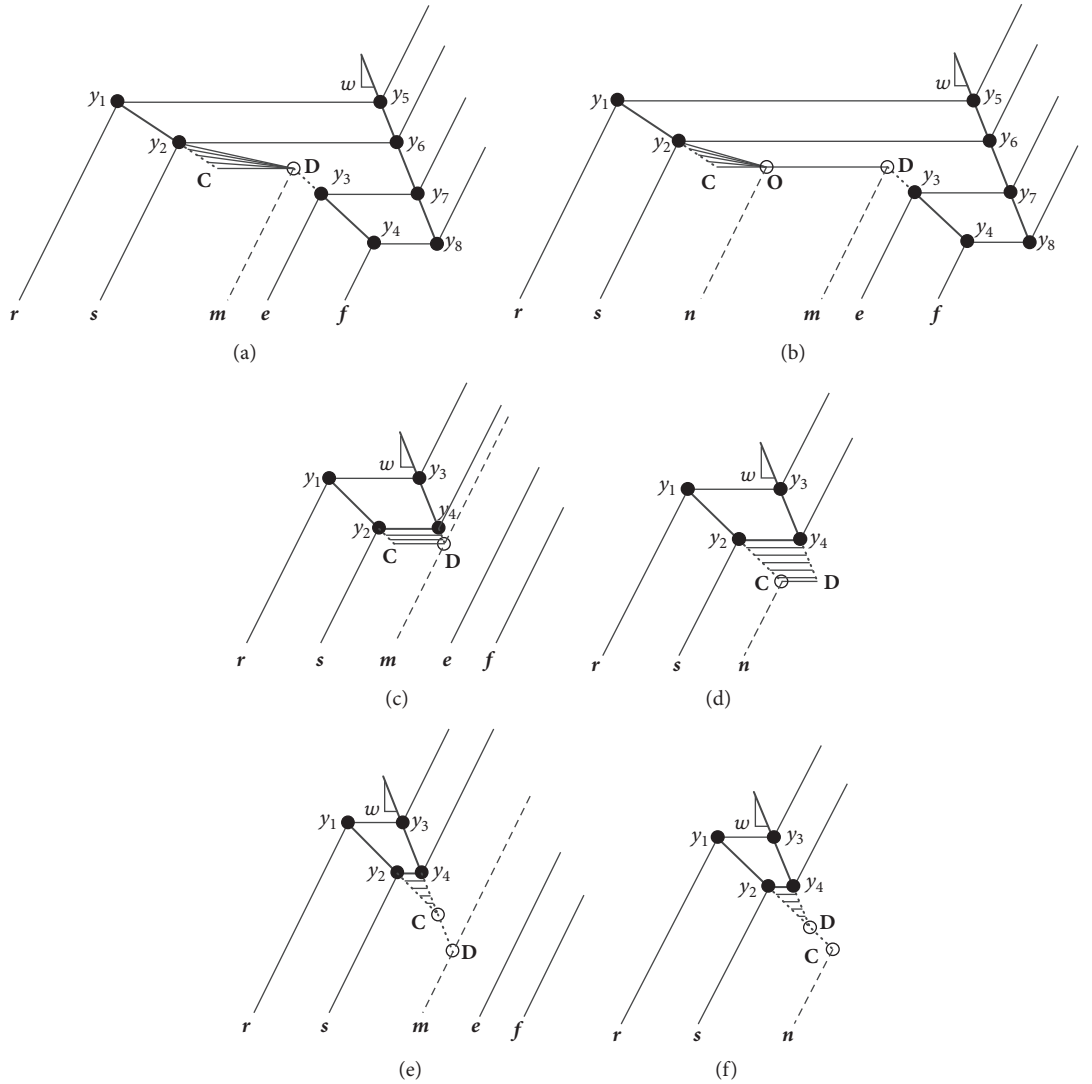


Figure 10: An illustration of handling missing data while constructing the shockwave queueing profiles using very limited mobile sensing data.

extreme case, while the real case can be any one between y_2y_4 and CD. Moreover, Figures 10(e) and 10(f) present two special but possible cases of Figures 10(c) and 10(d).

The combination of the cases in Figure 10 could very likely happen in practice. For instance, with Figures 10(a) and 10(d) in mind, Figure 11(a) illustrates the reconstruction of the shockwave profiles, under the assumption that each subflow contains only two CVs, e.g., r and s and c and d. More specifically, the shockwave forming curve $y_2\mathrm{D}y_3$ and the queue end EF can be fixed. Albeit not fully accurate, our simulation studies in Section 5 demonstrate that the inaccuracy is not essential as far as the total delay estimate is concerned (Theorem 6).

4.2.2. The Case of One Connected Vehicle in Each Subflow. At an early stage of the CV deployment, it may not be even practical to assume two CVs available in each flow from the upstream. So, it is necessary to also consider the case that only one CV is involved in some flows. Let us focus on Figure 11(b), where vehicle s and f are the only CVs within their respective flows. We again consider a virtual car n that may leave the upstream intersection at the end of the green phase that is associated with vehicle f. Then, we can fix point E with the shockwave dissolving speed w determined with y_6 and y_8. Furthermore, with the information of y_4 and the trajectory of virtual vehicle m that leaves at the start of the upstream green phase, we can determine point D. With D and y_2, we can determine point B.

Once the shockwave profile is fixed, the deterministic profile can also be plotted on the basis of Theorems 1, 2, and 4; see the illustrated correspondence between points A–E with points A′–E′ in Figure 11.

It should be emphasized that this work only requires the following information of the application site: density k_{jam} of the concerned road link, the start and end times of the signal phases at the upstream intersection and the subject intersection, and the position and speed information of CVs.

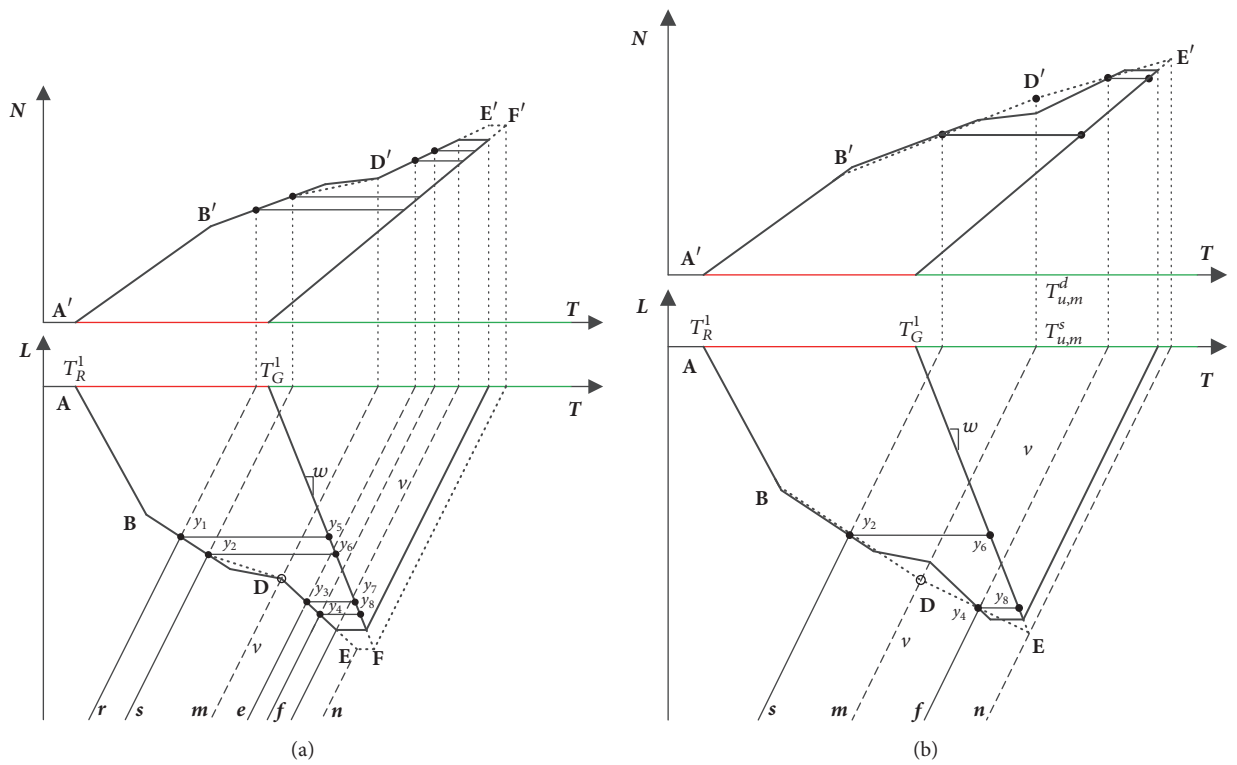


Figure 11: Further consideration for the queueing profile reconstruction.

5. Microscopic Simulation Evaluation

So far, we have discussed how to construct the shockwave and deterministic queueing profiles with very limited mobile sensing data. This section aims to evaluate the accuracy of the profile construction using the microscopic traffic simulator AIMSUN. Note that it makes little sense to do the evaluation on any performance index concerning individual cars, as the CV penetration rate is very low. Rather, we handle the evaluation on the basis of the total vertical and horizontal queue delays, which have been discussed with Theorem 6. More specifically, the ground truth is the sum of delays of all emulated individual vehicles over the simulation time horizon. The two intersections considered are 300 meters apart. The phase plan of the upstream intersection is given in Figure 12 with reference to Figure 8. The subject intersection at the downstream follows the same four-phase signal plan, with equal green and red phases of 30 s for the traffic flow from the concerned approach. The demands for the through, left-turn, and right-turn traffic at the upstream intersection (Figure 8) are 600 veh/h, 250 veh/h, and 150 veh/h. The jam density k_{jam} of the considered road link is determined with simulation to be 0.202 veh/m/lane. The simulation time horizon is 20 min. The simulation results are shown in Figure 7, but we focus on the period of the five most congested cycles from 270 s to 570 s.

By Theorem 6, we evaluate the accuracy of the profile construction based on the estimated total delay using both deterministic and shockwave profiles in comparison to the ground truth. The corresponding results are presented in Table 1. Clearly, the total delays determined with the two queueing profiles are quite close to the ground truth.

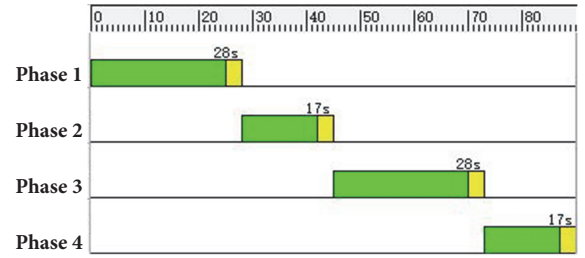


Figure 12: The phase plan for the upstream intersection.

As shown in Figure 7(a), 10 flows are involved. Except flow ⑨, each of those flows contains more than two vehicles. We assume each flow other than ⑨ contributes at most two CVs, and the only vehicle in flow ⑨ is a CV. We have then considered 10 different cases as displayed in Table 2. In each case, the blank space means the corresponding flow has two CVs while the cross means the corresponding flow has only one CV. From top to bottom in Table 2, the number of CVs is reduced by 1 each case. Finally, in case ⑩, every flow has only one CV. The 10 flows are of different volumes. The sequence of adding the cross signs is determined as follows: the less the volume that one flow has, the higher the possibility that it has only one CV.

Focusing on the five most congested cycles in Figure 7(a), Figures 13–15 present the queueing profiles constructed for all 10 cases aforementioned. The blue curves address the cases

Table 1: Comparison of total delay between the two queueing profiles in the case of 100% penetration rate.

	Delay of 100% CVs	Ratio by the ground truth
Deterministic	2405.00	98.52%
Shockwave	2341.72	95.93%
Ground truth	2441.20	100.00%

Table 2: Ten cases.

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
case 1									×	
case 2			×						×	
case 3			×						×	×
case 4			×			×			×	×
case 5	×		×			×			×	×
case 6	×		×			×	×		×	×
case 7	×		×	×		×	×		×	×
case 8	×		×	×	×	×	×		×	×
case 9	×	×	×	×	×	×	×		×	×
case 10	×	×	×	×	×	×	×	×	×	×

of 100% penetration rate, while the red curves correspond to the cases with at most two CVs in each subflow. It is noted that each horizontal pair of star signs represents a CV queueing. The corresponding penetration rate and estimation performance for each case are presented in Table 3.

Even in the extreme case ⑩, the performance is still acceptable. This indicates that the new technologies of mobile sensing have the potential of enabling even with a very low market penetration rate of connected vehicles a number of tasks that would otherwise be unpractical.

6. Conclusion

This paper addresses via analytical studies and microscopic simulations the lasting debates on the consistency between the deterministic/point/vertical queue model and the shockwave/physical/horizontal queue model, particularly with regard to oversaturated signal intersections. This paper also develops an efficient approach to the quasi-real-time reconstruction of the deterministic and shockwave queueing profiles using very limited mobile sensing data. Microscopic simulations with AIMSUN have demonstrated the efficiency of the approach as well as the analytical results obtained.

Appendix

A. The Case of a Constant Upstream Demand

For the convenience of mathematical derivation, the proof is presented in the sequence of Theorems 1, 2-(1), and 4, Corollary 5, Theorem 2-(2), Corollary 3, and Theorem 6.

Proof of Theorem 1.

(1) $\theta = 1$. Note in Figure 2 that the $N^1_{\max}$th vehicle in the deterministic profile is exactly vehicle o involved with the shockwave profile. Then, with the deterministic profile,

$$N^1_{\max} = \left(T^1_G - T^1_R\right) \cdot \frac{q_s \cdot q_u}{q_s - q_u}$$
$$T^{d,1}_E = T^1_R + \frac{N^1_{\max}}{q_u} \tag{A.1}$$

Based on the shockwave profile,

$$L^1_{\max} = \left(T^1_G - T^1_R\right) \cdot \frac{|x| \cdot |w|}{|w| - |x|}$$
$$T^1_{\max} = T^1_R + \frac{L^1_{\max}}{|x|} \tag{A.2}$$
$$T^{s,1}_E = T^1_{\max} + \frac{L^1_{\max}}{v}$$

Then, it is easy to check $T^{s,1}_E = T^{d,1}_E$.

(2) $\theta = 2$. Note that the $N^2_{\max}$th vehicle in the deterministic profile and vehicle n in the shockwave profile are the same vehicle. Then, with the deterministic profile,

$$N^1_C = \left(T^2_R - T^1_G\right) \cdot q_s \tag{A.3}$$

$$N^2_{\max} = \left(\left(T^1_G - T^1_R\right) + \left(T^2_G - T^2_R\right)\right) \cdot \frac{q_s \cdot q_u}{q_s - q_u} \tag{A.4}$$

$$T^{d,2}_E = T^1_R + \frac{N^2_{\max}}{q_u} \tag{A.5}$$

Based on the shockwave profile,

$$L^2_{\min} = \left(T^{s,1}_E - T^2_R\right) \cdot \frac{|w| \cdot v}{|w| + v} \tag{A.6}$$

$$T^2_{\min} = T^2_R + \frac{L^2_{\min}}{|w|} \tag{A.7}$$

$$L^2_{\max} = L^2_{\min} + \left(T^2_G - T^2_R\right) \cdot \frac{|x| \cdot |w|}{|w| - |x|} \tag{A.8}$$

$$T^2_{\max} = T^2_G + \frac{L^2_{\max}}{|w|} \tag{A.9}$$

$$T^{s,2}_E = T^2_{\max} + \frac{L^2_{\max}}{v} \tag{A.10}$$

Then, it is ready to see $T^{s,2}_E = T^{d,2}_E$. □

Proof of Theorem 2-(1). Consider for instance vehicle i in Figure 2. If it kept moving at its original speed after it joins the horizontal queue, it would reach the stop line at time instant $T^s_{u,i}$. By the definition of vertical queue, $T^s_{u,i}$ is exactly the time that vehicle i joins the vertical queue, which is also time $T^d_{u,i}$

Table 3: Total delay comparison for ten cases.

Case	Number of CVs	Penetration rate	Total Delay (s)		Ratio by 100% CV data	
			Shockwave	Deterministic	Shockwave	Deterministic
1	19	22.62%	2412.16	2464.28	103.01%	102.46%
2	18	21.43%	2447.29	2508.28	104.51%	104.29%
3	17	20.24%	2429.25	2491.90	103.74%	103.61%
4	16	19.05%	2394.53	2477.85	102.26%	103.03%
5	15	17.86%	2415.03	2467.55	103.13%	102.60%
6	14	16.67%	2430.02	2497.02	103.77%	103.83%
7	13	15.48%	2419.52	2483.11	103.32%	103.25%
8	12	14.29%	2413.91	2409.39	103.08%	100.18%
9	11	13.10%	2405.28	2421.46	102.71%	100.68%
10	10	11.90%	2205.30	2204.71	94.17%	91.67%

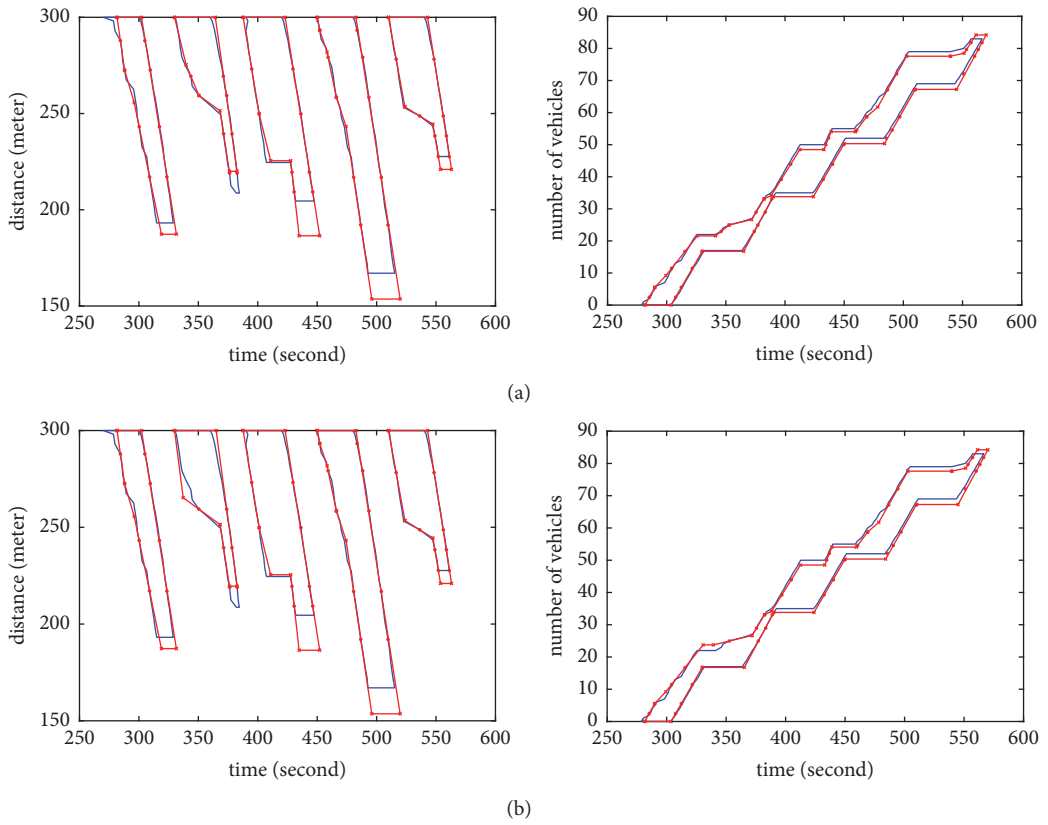


Figure 13: Reconstruction of queueing profiles for cases 1-2 in Table 2.

recorded on the arrival curve of the deterministic profile, i.e., $T^s_{u,i} = T^d_{u,i}$. Similarly, this conclusion also holds for vehicles p and m in Figure 2. In general, for any vehicle φ,

$$T^s_{u,\varphi} = T^d_{u,\varphi}. \tag{A.11}$$

□

Proof of Theorem 4. Without loss of generality, the proof is given with regard to vehicles i, p, and m in Figure 2.

Vehicle i. Based on the shockwave profile,

$$\begin{aligned} T^s_{u,i} &= T^1_R + \frac{L^1_i}{|x|} + \frac{L^1_i}{v} \\ &= T^1_R + \frac{L^1_i}{q_u} \cdot k_{\text{jam}} \end{aligned} \tag{A.12}$$

Based on the deterministic profile,

$$T^d_{u,i} = T^1_R + \frac{N_i}{q_u} \tag{A.13}$$

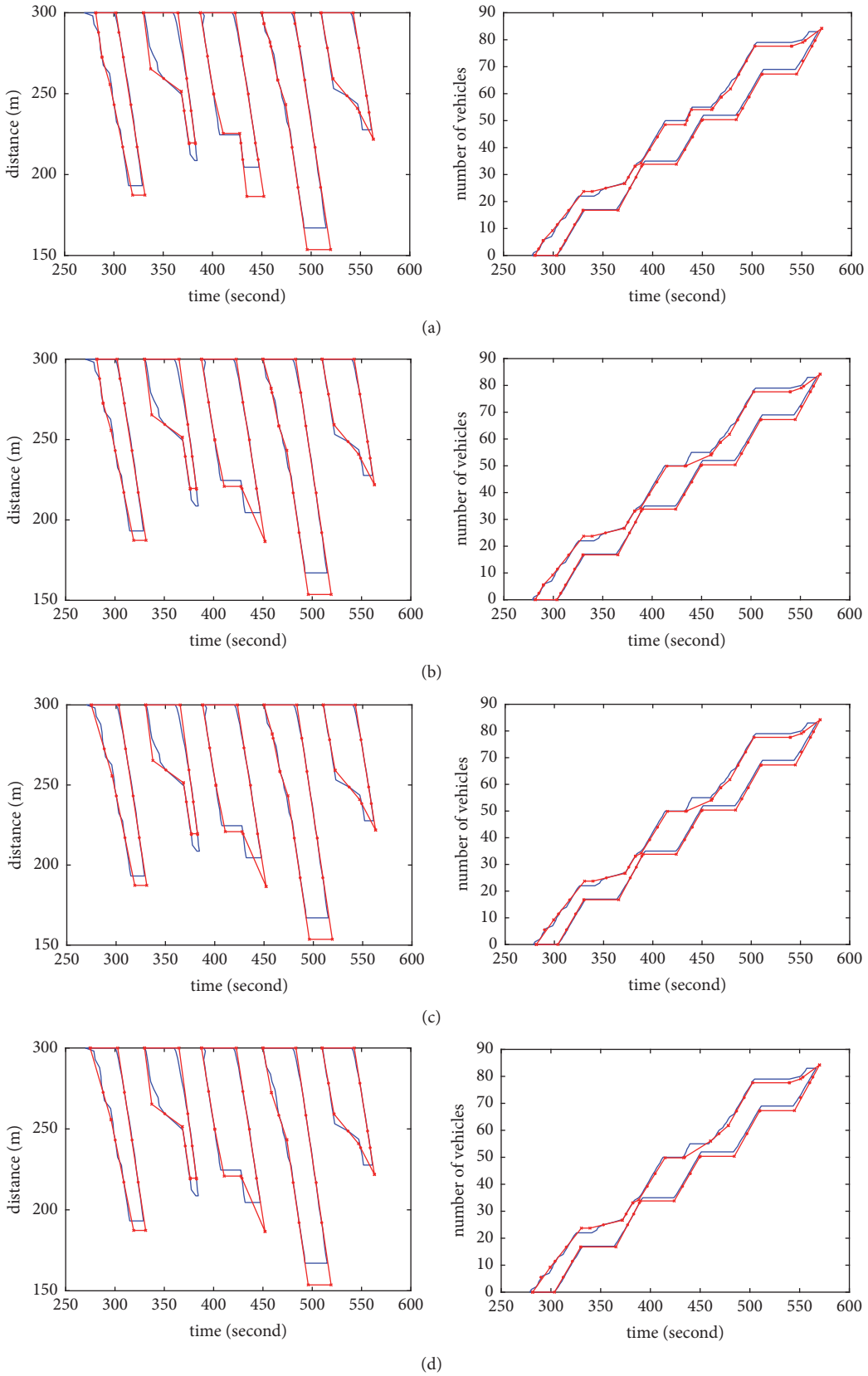


Figure 14: Reconstruction of queueing profiles for cases 3–6 in Table 2.

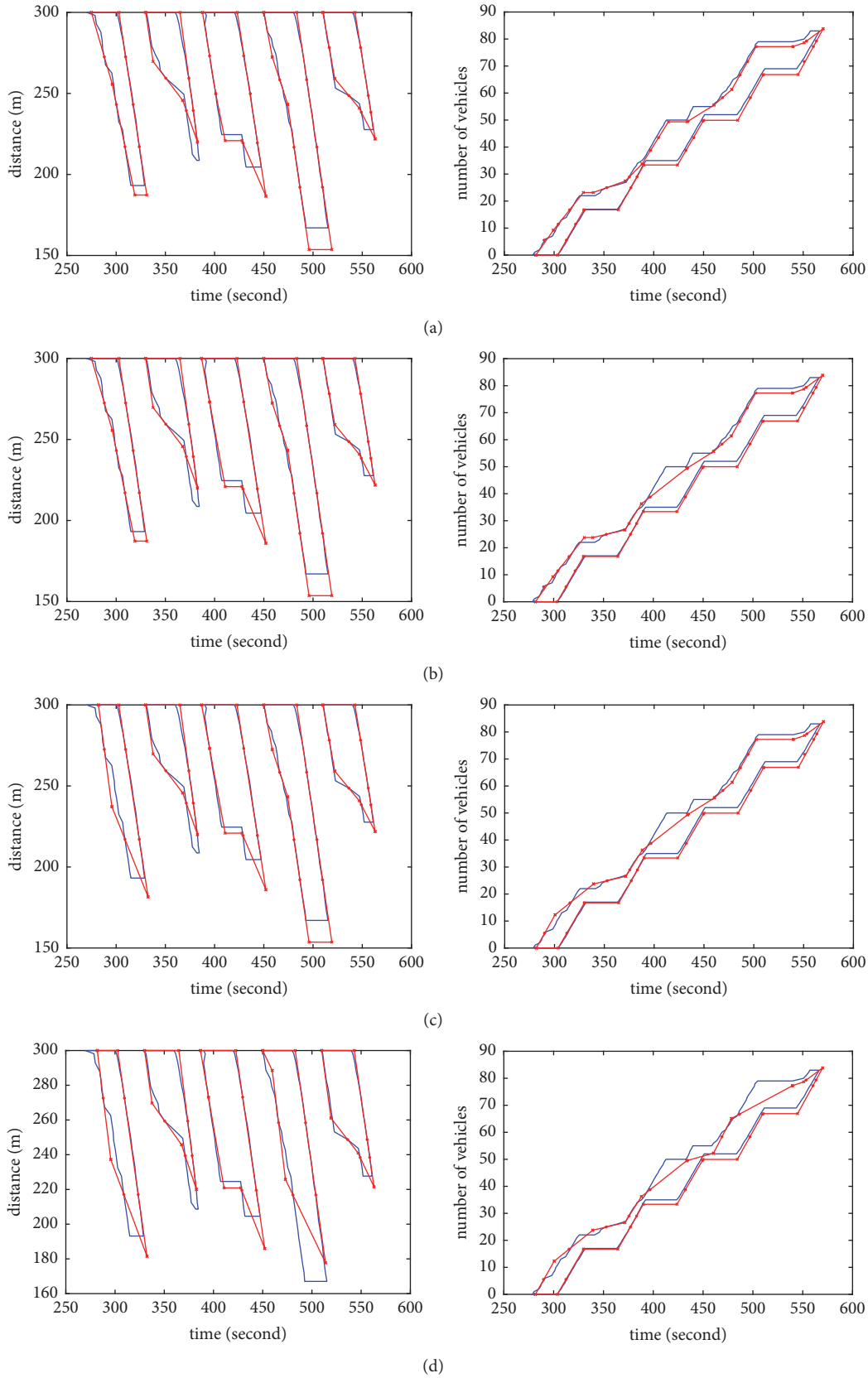


Figure 15: Reconstruction of queueing profiles for cases 7–10 in Table 2.

Thus, we have, by (A.11),

$$N_i = L_i^1 \cdot k_{\text{jam}} \tag{A.14}$$

Vehicle p. Concerning the first cycle, it can be similarly verified:

$$N_p = L_p^1 \cdot k_{\text{jam}}, \tag{A.15}$$

For the second cycle, based on the shockwave profile in Figure 2,

$$T_G^1 + \frac{L_p^1}{|w|} + \frac{L_p^1}{v} = T_R^2 + \frac{L_p^2}{|w|} + \frac{L_p^2}{v} \tag{A.16}$$

Then,

$$L_p^1 = \frac{N_C^1}{k_{\text{jam}}} + L_p^2 \tag{A.17}$$

Substituting (A.17) into (A.15) leads to

$$N_p - N_C^1 = L_p^2 \cdot k_{\text{jam}} \tag{A.18}$$

Vehicle m. Based on the shockwave profile and (A.6),

$$\begin{aligned} T_{u,m}^s &= T_R^2 + \frac{L_{\min}^2}{|w|} + \frac{L_m^2 - L_{\min}^2}{|x|} + \frac{L_m^2}{v} \\ &= T_R^1 + \frac{N_C^1}{q_u} + \frac{L_m^2 \cdot k_{\text{jam}}}{q_u} \end{aligned} \tag{A.19}$$

With the deterministic diagram,

$$T_{u,m}^d = T_R^1 + \frac{N_m}{q_u} \tag{A.20}$$

Then, by (A.11),

$$N_m - N_C^1 = L_m^2 \cdot k_{\text{jam}} \tag{A.21}$$

□

Proof of Corollary 5. For vehicle p queueing twice as illustrated in Figure 2, we can see by (A.3) and (A.17) that $L_p^2 = L_p^1 - ((T_R^2 - T_G^1) \cdot q_s)/k_{\text{jam}}$. □

Proof of Theorem 2-(2).

Vehicle i. Based on the shockwave profile in Figure 2,

$$\begin{aligned} T_{s,i}^s &= T_G^1 + \frac{L_i}{|w|} + \frac{L_i}{v} \\ &= T_G^1 + \frac{L_i}{q_s} \cdot k_{\text{jam}} \end{aligned} \tag{A.22}$$

With the deterministic profile,

$$T_{s,i}^d = T_G^1 + \frac{N_i}{q_s} \tag{A.23}$$

By (A.22), (A.23), and (A.14), we see that $T_{s,i}^d = T_{s,i}^s$ for any vehicle i.

Vehicle p. Similarly, it can be verified that $T_{s,p}^{d,1} = T_{s,p}^{s,1}$ for any vehicle p.

Based on the shockwave profile,

$$\begin{aligned} T_{s,p}^{s,2} &= T_G^2 + \frac{L_p^2}{|w|} + \frac{L_p^2}{v} \\ &= T_G^2 + \frac{L_p^2}{q_s} \cdot k_{\text{jam}} \end{aligned} \tag{A.24}$$

With the deterministic profile,

$$T_{s,p}^{d,2} = T_G^2 + \frac{N_p - N_C^1}{q_s} \tag{A.25}$$

Then, we know from (A.24), (A.25), and (A.18) that $T_{s,p}^{d,2} = T_{s,p}^{s,2}$ for any vehicle p.

Vehicle m. Based on the shockwave profile,

$$\begin{aligned} T_{s,m}^s &= T_G^2 + \frac{L_m^2}{|w|} + \frac{L_m^2}{v} \\ &= T_G^2 + \frac{L_m^2}{q_s} \cdot k_{\text{jam}} \end{aligned} \tag{A.26}$$

With the deterministic profile,

$$T_{s,p}^d = T_G^2 + \frac{N_m - N_C^1}{q_s} \tag{A.27}$$

Via (A.26), (A.27), and (A.21), we have $T_{s,m}^d = T_{s,m}^s$ for any vehicle m. □

Proof of Corollary 3. Based on the definitions of queueing delay, $d_\varphi^d = T_{s,\varphi}^d - T_{u,\varphi}^d$, $d_\varphi^s = T_{s,\varphi}^s - T_{u,\varphi}^s$ (if φ is involved with one cycle) or $d_\varphi^s = T_{s,\varphi}^{s,2} - T_{u,\varphi}^s$ (if φ is involved with two cycles). Then, it is straightforward with Theorem 2 that Corollary 3 holds. □

Proof of Theorem 6. As illustrated with Figure 16, there exists one-to-one correspondence between the deterministic and shockwave profiles in terms of vehicle delays. In what follows, we use subscript "H" to represent the horizontal queue and subscript "V" to represent the vertical queue.

Case ①. With the deterministic profile in Figure 16,

$$S_{V,1} = \frac{1}{2} \cdot \left(T_G^1 - T_R^1\right) \cdot N_{\max}^1 \tag{A.28}$$

Based on the shockwave profile,

$$S_{H,1} = \frac{1}{2} \cdot \left(T_G^1 - T_R^1\right) \cdot L_{\max}^1 \tag{A.29}$$

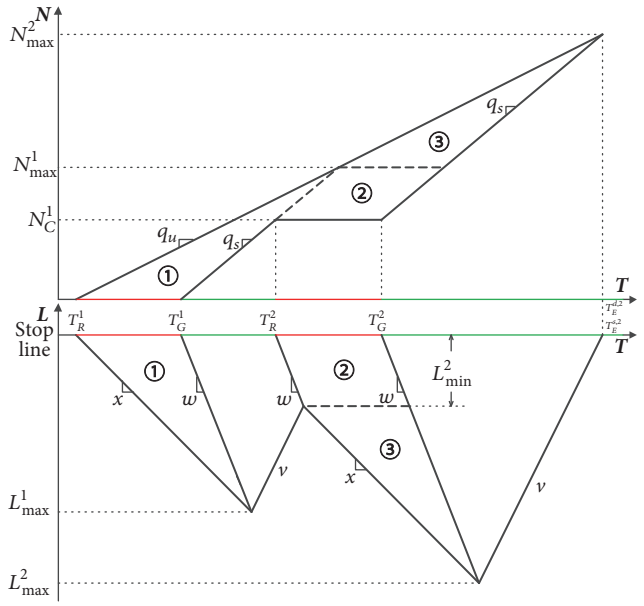


FIGURE 16: Deterministic and shockwave profiles with their total delay connection.

Based on Theorem 4, we have $N^1_{max} = L^1_{max} \cdot k_{jam}$ for vehicle o (in Figure 2); then $S_{V,1} = S_{H,1} \cdot k_{jam}$.

Case ②

$$\begin{aligned} S_{V,2} &= \left(T^2_G - T^2_R\right) \cdot \left(N^1_{max} - N^1_C\right) \\ S_{H,2} &= \left(T^2_G - T^2_R\right) \cdot L^2_{min} \end{aligned} \tag{A.30}$$

Again, based on Theorem 4 and Figure 2, $N^1_{max} - N^1_C = L^2_{min} \cdot k_{jam}$ for vehicle o (in Figure 2); we have $S_{V,2} = S_{H,2} \cdot k_{jam}$.

Case ③

$$\begin{aligned} S_{V,3} &= \left(T^2_G - T^2_R\right) \cdot \left(N^2_{max} - N^1_{max}\right) \\ S_{H,3} &= \left(T^2_G - T^2_R\right) \cdot \left(L^2_{max} - L^2_{min}\right) \end{aligned} \tag{A.31}$$

Based on Theorem 4, $N^1_{max} - N^1_C = L^2_{min} \cdot k_{jam}$ and $N^2_{max} - N^1_C = L^2_{max} \cdot k_{jam}$; then we have $S_{V,3} = S_{H,3} \cdot k_{jam}$.

Since $S_V = S_{V,1} + S_{V,2} + S_{V,3}$ and $S_H = S_{H,1} + S_{H,2} + S_{H,3}$, $S_V = S_H \cdot k_{jam}$. □

B. The Case of a Varying Upstream Demand

Theorems 1, 2, 4, and 6 and Corollaries 3 and 5 still hold in this case. The corresponding proof is briefly presented with reference to Figure 6.

Proof of Theorem 1. Regardless of traffic demands, the departure curve in either profile is a straight line. With this in mind, the total number of vehicles served during the two cycles (passing the stop line) in the shockwave profile of Figure 6 can be derived as $N_H = ((T^2_R - T^1_G) + (T^{s,2}_E - T^2_G)) \cdot q_s$, under the assumption that vehicle queues are discharged during the green phase at the full capacity q_s of the intersection. This number can also be checked with the deterministic profile, and $N_V = ((T^2_R - T^1_G) + (T^{d,2}_E - T^2_G)) \cdot q_s$. Whichever of the models is used, we must have $N_V = N_H$. Thus, it can be easily verified that $T^{s,2}_E = T^{d,2}_E$. Similarly, we also have $T^{s,1}_E = T^{d,1}_E$. □

As shown in Figure 6, vehicles that join the horizontal queue over section AB can be regarded as those of a constant upstream demand. Thus, Theorems 2 and 4 apply to all these vehicles, including vehicle i. Then, the question is if Theorems 2 and 4 also apply to all vehicles that arrive over section BC provided that Theorems 2 and 4 apply to vehicle i at point B. If it is true, Theorems 2 and 4 can be proved with mathematical induction for the deterministic and shockwave profiles in Figure 6, as long as the arrival curve in either profile is piecewise linear.

Without loss of generality, the proof is given below with respect to vehicles r and p in Figure 6. The proof process is essentially the same as that in the case of a constant demand. First, for vehicle i, we have, by Theorems 2 and 4,

$$T^d_{u,i} = T^s_{u,i} \tag{B.1}$$

$$N_i = L^1_i \cdot k_{jam} \tag{B.2}$$

Proof of Theorem 2-(1). Irrespective of the demand condition, this holds following the definitions of vertical and horizontal queues. □

Proof of Theorem 4. Without loss of generality, it suffices to check the cases of vehicles r and p.

Vehicle r. Based on the deterministic profile in Figure 6,

$$T^d_{u,r} = T^d_{u,i} + \frac{N_r - N_i}{q_{BC}}, \tag{B.3}$$

where q_{BC} denotes the upstream arrival flow for section BC.

Based on the shockwave profile,

$$\begin{aligned} T^s_{u,r} &= T_{a,i} + \frac{L^1_r - L^1_i}{|x_{BC}|} \\ &\quad + \frac{L^1_r}{v} \left(T_{a,i} \text{ is defined in Figure 2}\right) \\ &= T_{a,i} + \frac{L^1_i}{v} + \frac{L^1_r - L^1_i}{|x_{BC}|} + \frac{L^1_r - L^1_i}{v} \\ &= T^s_{u,i} + \frac{L^1_r - L^1_i}{q_{BC}/\left(k_{jam} - k_2\right)} + \frac{L^1_r - L^1_i}{q_{BC}/k_2} \\ &= T^s_{u,i} + \frac{L^1_r - L^1_i}{q_{BC}} \cdot k_{jam} \end{aligned} \tag{B.4}$$

Then, we have by (B.1)

$$N_r - N_i = \left(L^1_i - L^1_r\right) \cdot k_{jam} \tag{B.5}$$

Also, with (B.2) for vehicle i, it is straightforward to get

$$N_r = L_r^1 \cdot k_{\text{jam}} \tag{B.6}$$

Vehicle p. For the first cycle, like (B.6), it can be similarly verified that

$$N_p = L_p^1 \cdot k_{\text{jam}} \tag{B.7}$$

For the second cycle, based on the shockwave profile in Figure 6,

$$T_G^1 + \frac{L_p^1}{|w|} + \frac{L_p^1}{v} = T_R^2 + \frac{L_p^2}{|w|} + \frac{L_p^2}{v} \tag{B.8}$$

Then, by (A.3)

$$L_p^1 = \frac{N_C^1}{k_{\text{jam}}} + L_p^2 \tag{B.9}$$

Substituting (B.9) into (B.7) leads to

$$N_p - N_C^1 = L_p^2 \cdot k_{\text{jam}} \tag{B.10}$$

Thus, Theorem 4 holds for any vehicle p. □

Proof of Corollary 5. For vehicle p in Figure 6, via (A.3) and (B.9), $L_p^2 = L_p^1 - ((T_R^2 - T_G^1) \cdot q_s)/k_{\text{jam}}$. □

Proof of Theorem 2-(2). Again, it suffices to check vehicles r and p.

Vehicle r. With the deterministic profile,

$$T_{s,r}^d = T_G^1 + \frac{N_r}{q_s} \tag{B.11}$$

And by the shockwave profile,

$$\begin{aligned} T_{s,r}^s &= T_G^1 + \frac{L_r^1}{|w|} + \frac{L_r^1}{v} \\ &= T_G^1 + \frac{L_r^1}{q_s} \cdot k_{\text{jam}} \end{aligned} \tag{B.12}$$

By (B.11) and (B.12) as well as (B.6), we see that $T_{s,r}^d = T_{s,r}^s$ for any vehicle r.

Vehicle p. Similarly, it can be verified that $T_{s,p}^{d,1} = T_{s,p}^{s,1}$ for any vehicle p.

Based on the shockwave profile in Figure 6,

$$\begin{aligned} T_{s,p}^{s,2} &= T_G^2 + \frac{L_p^2}{|w|} + \frac{L_p^2}{v} \\ &= T_G^2 + \frac{L_p^2}{q_s} \cdot k_{\text{jam}} \end{aligned} \tag{B.13}$$

Based on the deterministic profile in Figure 6,

$$T_{s,p}^{d,2} = T_G^2 + \frac{N_p - N_C^1}{q_s} \tag{B.14}$$

Then, we know from (B.10), (B.13), and (B.14) that $T_{s,p}^{d,2} = T_{s,p}^{s,2}$ for any vehicle p. □

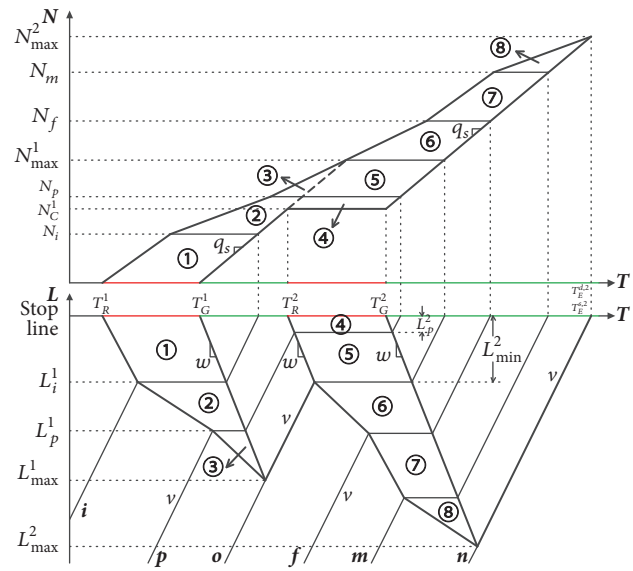


Figure 17: Deterministic and shockwave queueing profiles with their total delay correlation under a varying traffic demand.

As discussed above, Theorems 2 and 4 hold for any vehicle over BC provided that the theorems hold for vehicle i at point B. Similarly, it can also be proved that Theorems 2 and 4 hold for any vehicle over CD provided that the theorems hold for vehicle p at point C. As such, it can be proved that Theorems 2 and 4 hold for the whole shockwave and deterministic queueing profiles as long as the upstream demand is such that both queue forming curves are piecewise linear.

Proof of Corollary 3. The proof is straightforward. □

Proof of Theorem 6. As displayed with Figure 17, there exists one-to-one correspondence between the deterministic and shockwave profiles in terms of delay. The proof is similar to that for Theorem 6 in Appendix A. □

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Yongyang Liu and Jingqiu Guo contributed equally to this work and share the first authorship.

Acknowledgments

The research reported in this paper was supported in part by the Zhejiang Qianren Program (4013-2018) and the National Natural Science Foundation of China (51478428; 71771200).

References

[1] K. Moskowitz and L. Newman, "Notes on freeway capacity," Highway Research Board 27, 1963.

[2] D. C. Gazis and R. B. Potts, "The over-saturated intersection," in *Proceedings of the 2nd International Symposium on the Theory of Road Traffic Flow*, pp. 221–237, 1965.

[3] C. F. Daganzo, *Fundamentals of Transportation and Traffic Operations*, Pergamon, Oxford, UK, 1997.

[4] M. J. Lighthill and G. B. Whitham, "On kinematic waves. II. A theory of traffic flow on long crowded roads," *Proceedings of the Royal Society A Mathematical, Physical and Engineering Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[5] P. I. Richards, "Shock waves on the highway," *Operations Research*, vol. 4, no. 1, pp. 42–51, 1956.

[6] W. R. McShane and R. P. Roess, *Traffic Engineering*, Prentice Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 1998.

[7] Y. Makigami, G. F. Newell, and R. Rothery, "Three-dimensional representation of traffic flow," *Transportation Science*, vol. 5, no. 3, pp. 302–313, 1971.

[8] S. C. Wirasinghe, "Determination of traffic delays from shockwave analysis," *Transportation Research*, vol. 12, no. 5, pp. 343–348, 1978.

[9] P. G. Michalopoulos and V. B. Pisharody, "Derivation of delays based on improved macroscopic traffic models," *Transportation Research Part B: Methodological*, vol. 15, no. 5, pp. 299–317, 1981.

[10] C. F. Daganzo, "Derivation of delays based on input-output analysis," *Transportation Research Part A: General*, vol. 17, no. 5, pp. 341-342, 1983.

[11] H.-C. Chin, "A reexamination of the analysis of freeway bottlenecks," *ITE Journal (Institute of Transportation Engineers)*, vol. 66, no. 1, pp. 30–35, 1996.

[12] D. H. Nam and D. R. Drew, "Analyzing freeway traffic under congestion: traffic dynamics approach," *Journal of Transportation Engineering*, vol. 124, no. 3, pp. 208–212, 1998.

[13] T. W. Lawson, D. J. Lovell, and C. F. Daganzo, "Using input-output diagram to determine spatial and temporal extents of a queue upstream of a bottleneck," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1572, pp. 140–147, 1997.

[14] A. L. Erera, T. W. Lawson, and C. F. Daganzo, "Simple, generalized method for analysis of traffic queue upstream of a bottleneck," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1646, pp. 132–140, 1998.

[15] D. J. Lovell and J. R. Windover, "Analyzing freeway traffic under congestion: Traffic dynamics approach," *Journal of Transportation Engineering*, vol. 125, no. 4, pp. 373–375, 1999.

[16] H. Rakha and W. Zhang, "Consistency of shock-wave and queuing theory procedures for analysis of roadway bottlenecks," in *Proceedings of the 84th Annual Meeting of the Transportation Research Board*, Washington, DC, USA, 2005.

[17] P. Yi, Z.-Z. Tian, and Q. Zhao, "Consistency of input-output model and shockwave analysis in queue and delay estimations," *Journal of Transportation Systems Engineering and Information Technology*, vol. 8, no. 6, pp. 146–152, 2008.

[18] H. X. Liu, X. W. Wu, Ma., and H. Hu, "Real-time queue length estimation for congested signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 4, pp. 412–427, 2009.

[19] A. Skabardonis and N. Geroliminis, "Real-time monitoring and control on signalized arterials," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 12, no. 2, pp. 64–74, 2008.

[20] A. Sharma, D. M. Bullock, and J. A. Bonneson, "Input-output and hybrid techniques for real-time prediction of delays and maximum queue length at a signalized intersection," *Journal of Transportation Research Record*, no. 2035, pp. 88–96, 2007.

[21] F. Dion, H. Rakha, and Y. Kang, "Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections," *Transportation Research Part B: Methodological*, vol. 38, no. 2, pp. 99–122, 2004.

[22] Z. Sun and J. Ban, "Vehicle trajectory reconstruction for signalized intersections using mobile traffic sensors," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 268–283, 2013.

[23] X.-Y. Lu and A. Skabardonis, "Freeway traffic shockwave analysis: exploring ngsim trajectory data," in *Proceedings of the 86th Annual Meeting of the Transportation Research Board*, Washington, DC, USA, 2007.

[24] P. Izadpanah, B. Hellinga, and L. Fu, "Automatic traffic shockwave identification using vehicles' trajectories," in *Proceedings of the 88th Annual Meeting of the Transportation Research Board*, 2009.

[25] Y. Cheng, X. Qin, J. Jin, and B. Ran, "An exploratory shockwave approach for signalized intersection performance measurements using probe trajectories," in *Proceedings of the Transportation Research Board 89th Annual Meeting*, 2010.

[26] K. Yang and M. Menendez, "A convex model for queue length estimation in a connected vehicle environment," in *Proceedings of the 96th Annual Meeting of Transportation Research Board*, 2017.

[27] X. Ban, R. Herring, P. Hao, and A. M. Bayen, "Delay pattern estimation for signalized intersections using sampled travel times," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2130, pp. 109–119, 2009.

[28] X. J. Ban, P. Hao, and Z. Sun, "Real time queue length estimation for signalized intersections using travel times from mobile sensors," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1133–1156, 2011.

[29] P. Hao and X. Ban, "Long queue estimation for signalized intersections using mobile data," *Transportation Research Part B: Methodological*, vol. 82, pp. 54–73, 2015.

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 5728042, 11 pages
https://doi.org/10.1155/2018/5728042

Research Article

High-Speed Data-Driven Methodology for Real-Time Traffic Flow Predictions: Practical Applications of ITS

Hyun-ho Chang[1] **and Byoung-jo Yoon**[2]

[1]*School of Environmental Studies, Seoul National University, Seoul, Republic of Korea*
[2]*Department of Urban Engineering, Incheon National University, Incheon, Republic of Korea*

Correspondence should be addressed to Byoung-jo Yoon; bjyoon63@inu.ac.kr

Received 18 August 2017; Revised 28 February 2018; Accepted 8 March 2018; Published 24 April 2018

Academic Editor: Anastasios Kouvelas

Despite the achievements of academic research on data-driven *k*-nearest neighbour nonparametric regression (KNN-NPR), the low-speed computational capability of the KNN-NPR method, which can occur during searches involving enormous amounts of historical data, remains a major obstacle to improvements of real-system applications. To overcome this critical issue successfully, a high-speed KNN-NPR framework, capable of generating short-term traffic volume predictions, is proposed in this study. The proposed method is based on a two-step search algorithm, which has the two roles of building promising candidates for input data during nonprediction times and identifying decision-making input data for instantaneous predictions at the prediction point. To prove the efficacy of the proposed model, an experimental test was conducted with large-size traffic volume data. It was found that the performance of the model not only at least equals that of linear-search-based KNN-NPR in terms of prediction accuracy, but also shows a substantially reduced execution time in approximating real-time applications. This result suggests that the proposed algorithm can be also effectively employed as a preprocess to select useful past cases for advanced learning-based forecasting models.

1. Introduction

In intelligent transport systems (ITS), vehicular traffic variables such as volumes, speeds, travel times, and occupancies collected by traffic monitoring devices are not current but are data which were temporally collected previously. To produce accurate real-time or future traffic information, a prediction system is essential as one of the subsystems of ITS, and it is natural for the core of the system to be a reliable prediction model. As such, short-term prediction modelling is a crucial and attractive topic in ITS research.

In modern ITS, the popular introduction of advanced data management systems has made tremendous quantities of traffic variable data available. The wealth of traffic data, in turn, has rendered promising opportunities to data-driven forecasting approaches, one of which is data-driven *k*-nearest neighbour nonparametric regression (KNN-NPR), as KNN-NPR has its roots in strong pattern recognition, that is, the diversity of the patterns included in rich historical data [1]. KNN-NPR is also called the *k*-nearest neighbours (KNN) method [2], as it essentially employs the *k*-NN strategy to mine the neighbourhoods directly, that is, *k* cases similar to a current case, which in turn is used to understand the temporal evolution of current states [1–3]. Despite the academic and practical advantages [3] of KNN predictions as discussed in the literature review here, a chronic weakness of data-driven KNN in time-critical systems has been the long execution time [4, 5], mainly required to search for past observations, that is, KNN, which are similar to current observations. This computational issue of KNN should therefore be addressed thoroughly before real-world applications can be realized.

To overcome the shortcomings of data-driven KNN as mentioned above, a high-speed KNN framework for predicting short-term traffic flows is proposed in this article. The method is developed based on a two-step data search algorithm, which is designated for two separate roles to find candidate input data similar to the current state included in historical data during nonprediction times and then

instantaneously to identify decision-making input data for predictions from the candidate input data at the forecasting point. The performances of the prediction algorithm are proven through a prediction simulation with real-world motorway traffic volume data, and certain findings are discussed from academic and practical perspectives.

2. Backgrounds

There is a predominant notion that the evolutions of time-series traffic flow states naturally are chaotic systems [1, 6] in which the development of states is sensitively determined given certain initial conditions. KNN has its theoretical foundation in chaos theory in the area of time-series pattern recognition [1, 2], as follows: the decision with regard to future states made by a KNN predictor has the asymptotically minimum risk level, as $n \to \infty$, $k \to \infty$ with $k/n \to 0$, where n and k are possible patterns included in historical data and nearest patterns (similar) to the current time-series state, respectively. Particularly, the KNN approach basically relies on the bulk of knowledge [3] included in the historical data to understand the relationship between the input and the output variables [2, 4] without any statistical assumptions [2], as opposed to statistical knowledge inferred by man-made artificial formulae [2–4].

Due to its theoretical and practical advantages, KNN has developed into a promising approach for ITS forecast modelling. (Note that research on traffic volume predictions based on data-driven KNN is the focus here; several wide-ranging reviews [7, 8] of various forecasting approaches for traffic variables are recommended for more interested readers.) The KNN strategy for nonlinear clustering was extended to a time-series analysis as a data-driven KNN method [2], after which it was also extended to multi-interval forecasting [1, 3, 9, 10] and improved for multivariate estimations [11, 12]. To enhance the prediction accuracy, academic efforts are still realized with sophisticated similarity measures [9, 13] and/or updated forecasting functions [1, 2, 13]. It was also reported in comparative studies [1–3, 9, 12] that the performances of KNN-based methods in terms of prediction reliability are at least comparable to those of parametric and/or nonlinear models.

Despite these achievements, the weakness of data-driven KNN has been its slow execution time [4, 5] from the perspective of time-critical systems such as dynamic ITS information systems. A majority of the run time is spent searching for past cases (similar to the current case) included in rich historical data, as a linear search is essential to build the best group of past cases. To address this problem, several techniques to reduce the search time can be categorized into two approaches: advanced search technology [4] and data-segmentation methods [3, 5, 10, 12]. As for advanced search technology, an imprecise computational method based on approximate nearest neighbour (ANN) searching supported by an advanced data management system (ADMS) was reported in [4], where searching time was reduced to the degree of 44.0~67% with an acceptable level of +1% prediction error. However, additional academic studies supported by ADMS have not been reported in the area of ITS forecasting, since ADMS that is composed of advanced hardware and software devices is deeply related to tight budgets, which is another hindrance in practice. The data-segmentation method narrows the entire historical data down to useful data with the assumption that the temporal variation of traffic flow is recurrent within a time span of a day, an hour, or even several minutes. This can be efficacy, as searching time is proportional to the size of searched data in the case of a linear search. A useful sector of a historical database is time-dependently constrained within a fixed window of the present time interval $\pm\beta$ [3, 12] or the type of day [10] to reduce the load when searching. The β-based methods reduce a size of historical data to $(2\beta + 1)/S_n$, where S_n is the number of time sequences in a day (see Figure 1(a)). These fixed-window methods are useful for searching time but they cannot yield reliable predictions. To handle this problem effectively, a flexible segmentation method to preset a useful segment of the database was proposed in [5]. The flexible segmentation is preset through a sort of KNN prediction simulation using similarity ranking and prediction-error ranking, and each segment for a time interval has an s-size array that includes a ranking score (z, 0.0~1.0). If a ranking score (z_i) for s_i ($s_i \in S$) is more than α value (0.0~1.0), then each historical data corresponding to the s_i sequence is searched in the searching step of a standard KNN (SKNN) (see Figure 1(b)). In addition, the procedure to predetermine flexible segmentation is very time-expensive and should be executed offline to update segmentation information on a monthly or weekly basis. Most importantly, the data-segmentation approach cannot guarantee the reliability of prediction, when the temporal variation of traffic volume is nonrecurrent (see Figure 2(b)). The temporal pattern of traffic flow deviates from the notions of periodicity (monthly, daily, or hourly) [14]. Despite these efforts, it is clear that the execution time of the data-driven KNN forecasting algorithm is not comparable to those of high-speed real-time models. In addition, a KNN algorithm has still not been reported, which can dynamically and effectively predetermine a small member of promising past cases by reflecting current traffic flow states under the conditions of conventional ITS systems.

Based on the literature review, the performance of KNN has reached an acceptable level of prediction accuracy in virtue of academic efforts toward refined modelling. However, it appears that the execution time of the data-driven KNN algorithm had yet to progress sufficiently. Most importantly, the framework of the KNN forecasting algorithm inevitably includes a search process, which mainly contributes to its long execution time. For this reason, the KNN forecasting procedure can be a bottleneck in dynamic information flows in conventional ITS systems that are not supported with any advanced data management or search technologies.

Unquestionably, the "bigger data and slower running" problem associated with KNN-based forecasting remains an ongoing issue to be addressed successfully and urgently. To make matters worse, the volume of historical data available continues to grow in modern ITS. Hence, a high-speed framework for KNN algorithms is necessary, and

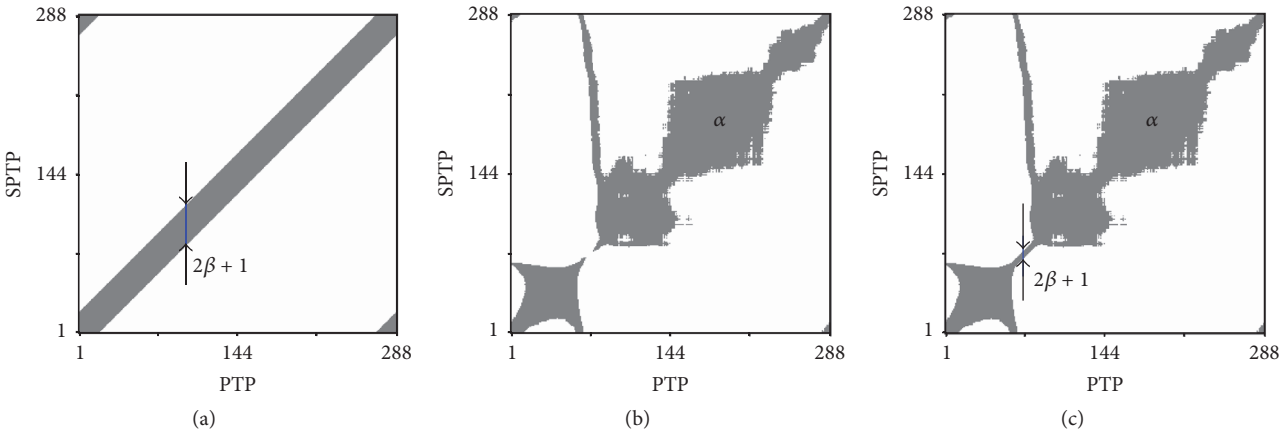


FIGURE 1: Static data segments used for forecasting. (a) KNN-D, (b) KNN-U, and (c) KNN-UD. Note: gray area is used segment, PTP (prediction time point, 5 min), and SPTP (selected past time point, 5 min).

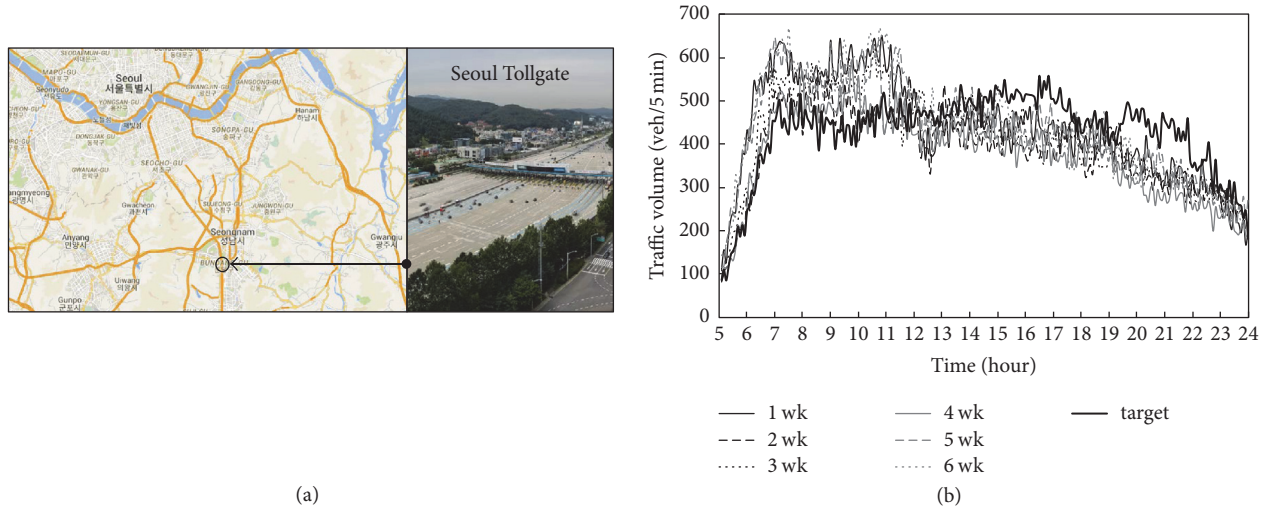


FIGURE 2: Information of target traffic volume data. (a) Test bed (source: Google Maps) and (b) temporal state evolution of the target data.

this represents another challenge. In addition, a high-speed process to search and identify similar cases from tremendous historical data is necessary as a learning step of advanced forecasting models such as a support vector machine or deep learning.

3. High-Speed Framework for KNN Prediction

3.1. Concept. The aim of this study is to develop a high-speed framework of KNN-based prediction while considering partial current traffic flow states in order to extremely speed up a KNN prediction method and guarantee its prediction accuracy. The slow running issue of KNN algorithms can be surmounted simply by excluding the search process of KNN from the prediction algorithm. However, a new problem would be how to find and determine the potential NN to generate the forecast within a given time limit, $\varepsilon_r \rightarrow +0.0$, at prediction point t_p. Note that the structure of a KNN prediction algorithm is divided into two main parts: a search process to find KNN similar to the current states from the historical database and a forecast function to generate future states with the information of KNN.

A key clue to solving the problem exists in the basic premise of forecast modelling, which holds that the current temporal evolution of states is closely related to subsequent states in some way. Similarly, the relationship between the previous temporal development of states and the present state also follows this premise. Based on this self-evident relationship, incomplete current temporal states which do not include the present state can be effectively used to find promising instances of NN from a historical database with a similarity measure during nonprediction times (i.e., the length of the time interval (t_l)—ε_r), after which the exact NN to the complete current temporal states with acceptable differences, when the present state is collected, can then be instantly identified from a dataset of promising instances of NN during the process of the prediction algorithm within ε_r at t_p.

In this study, each of two-step search procedures above is combined with the framework of the KNN prediction algorithm. The algorithm minus the forecast generation

aspect is used as the first step of the search process to discern promising k-nearest neighbours (k_p-NN) from a historical database during the nonprediction time, after which the algorithm is again employed to select instantly the optimal k-nearest neighbours (k_o-NN) from the updated k_p-NN dataset using the second step of the search and then to generate future states at t_p, where $0 < k_o \leq k_p$, $k_p = f \times k_o$, and $f \geq 1.0$. Additionally, the k_p-NN dataset is updated simultaneously when the present state is collected. In this way, the search process of k-NN, the main cause of the long execution time of KNN, is significantly excluded from the algorithm, whereas the SKNN algorithm can be ensured as k_o-NN.

3.2. Fundamental Components. A KNN prediction model is typically composed of three fundamental components: (a) three state vectors (current, input, and output), (b) a similarity measure, and (c) a forecast function (FF) or algorithm. The high-speed framework proposed in this study was developed based on the general framework of the SKNN prediction algorithm. Under this consideration, various KNN models can be easily integrated into the high-speed framework. Each of the three components is defined as shown below, and the defined components are combined into the forecasting algorithm in Section 3.3.

In discrete systems, continuous time is divided by a fixed length of the time interval t_l; this is represented in the form of a time series, $[(t), (t-1), \ldots, (t-d)]$, at the present time interval (t) toward the past, where d is a suitable number of lags, that is, the embedding size. In our case, system states are traffic volume measurements, and the temporal development of the traffic volume states corresponding to the form of the time series is defined as the current state vector, $x_c(t)$, with

$$x_c(t) = \left[q(t), q(t-1), \ldots, q(t-d)\right]. \quad (1)$$

In this equation, $q(t)$ is the measurement during the time interval (t), $q(t-1)$ is the measurement during the time interval $(t-1)$, and so on. A one-dimensional univariate state vector is used in this study despite the fact that several types of state vectors are available for various purposes in KNN studies [1, 2].

To find and discern past cases similar to $x_c(t)$ from the historical database that consists of n past cases, the input state vector and the output state vector are necessary. The jth input state vector, $x_j(\tau)$, and output vector, o_j, at the past time interval (τ) are defined as (2) and (3), respectively, with $j \in n$. Additionally, the dimension of $x_j(\tau)$ is equal to that of $x_c(t)$.

$$x_j(\tau) = \left[q_j(\tau), q_j(\tau-1), \ldots, q_j(\tau-d)\right] \quad (2)$$

$$o_j = \left[u_j, q_j(\tau+1)\right]. \quad (3)$$

Here, τ is the (past) running-time index ($\tau < t-1$); o_j is associated with $x_j(\tau)$; u_j is the state distance, calculated with the Euclidean metric, between $x_c(t)$ and $x_j(\tau)$; and $q_j(\tau+1)$ is the past traffic volume record at time interval $(\tau+1)$.

To update and determine k-NN during the search process of the algorithm, various similarity measures can be used. In our case, the Euclidean distance is employed to calculate the degree of nearness between $x_c(t)$ and $x_j(\tau)$. Given the time-series dimension (l, d), $l = \{0, 1\}$, the state distance is defined as $u_j(l, d)$ with (4). In this way, the two types of state distances, that is, $u_j(1, d)$ and $u_j(0, d)$, are correspondingly utilized to determine the k_p-NN and k_o-NN datasets in the two-step search procedure of the proposed algorithm.

$$u_j(l, d) = \left[\sum_{s=l}^{d} \left|q_c(t-s) - q_j(\tau-s)\right|^2\right]^{1/2}. \quad (4)$$

The three defined vectors and the distance metric, which are combined with the search function, play a role in the building of the information of k-NN, which is composed of k [input → output] objects, where $k = \{k_p, k_o\}$, $0 < k_o \leq k_p$, $k_p = \alpha \times k_o$, and $\alpha \geq 1.0$. The k-objects are $x_i(\tau)$ and $o_i = [u_i(l, d), q_i(\tau+1)]$, where $i = 1, 2, \ldots, k$. To record the k-object information, a data structure is required. In this study, the data structure of k-NN is therefore defined for two functions: to record the k_p objects discerned in the historical database with $u_j(1, d)$ during nonprediction time and then instantaneously to determine the k_o-objects from the k_p-objects with $u_j(0, d)$ at $t_p$shown as follows.

Bilevel Data Structure for the Input and Output Vectors

$$\begin{array}{lccc} & \text{Input} & & \text{Output} \\ [i] & [k\text{-nearest neighbors}, x_i(\tau)] & & [k\text{-output vectors}, o_i] \\ 1 & [q_1(\tau), q_1(\tau-1), \ldots, q_1(\tau-d)] & \longrightarrow & [q_1(\tau+1), u_1(l,d)] \\ 2 & [q_2(\tau), q_2(\tau-1), \ldots, q_2(\tau-d)] & \longrightarrow & [q_2(\tau+1), u_2(l,d)] \\ \cdots & \cdots & & \cdots \\ k_o & [q_{k_o}(\tau), q_{k_o}(\tau-1), \ldots, q_{k_o}(\tau-d)] & \longrightarrow & [q_{k_o}(\tau+1), u_{k_o}(l,d)] \\ \cdots & \cdots & & \cdots \\ k_p & [q_{k_p}(\tau), q_{k_p}(\tau-1), \ldots, q_{k_p}(\tau-d)] & \longrightarrow & [q_{k_p}(\tau+1), u_{k_p}(l,d)]. \end{array} \quad (5)$$

Once the dataset of the k_o-objects has been selected, the last step is to estimate the future states with FF. A weighted function by the inverse of the state distance that was applied in previous studies [1–3, 10, 12] is improved and used as (6) with $l = 0$ in this work. This approach is based on the assumption that a more similar experience can lead to more reliable decisions about future uncertain states. Additionally, the FF with $l = 1$ can generate predictions when the present state, that is, $q(t)$, is not available at t_p due to a temporary failure or delay in the transmission of data from field-detector stations to a data center.

$$\widehat{q}(t+1) = \frac{\sum_{i=1}^{k_o} \left(q(\tau+1)/u_i(l,d)\right)}{\sum_{i=1}^{k_o} \left(1/u_i(l,d)\right)}. \tag{6}$$

Here, $\widehat{q}(t+1)$ is the estimated traffic volume for the future time interval $(t+1)$, $u_i(l,d) > 0.0$, and $l = \{0, 1\}$.

3.3. Prediction Algorithm. The three aforementioned components and the input-and-output data structure are integrated into the two-step search (S2S) KNN algorithm as shown in Algorithm 1. The KNN-S2S algorithm is designed for two main purposes: to build k_p-NN (i.e., k_p-objects) during the nonprediction time between the previous $t_p + \varepsilon_r$ and t_p and then to determine k_o-NN (i.e., k_o-objects) and generate $\widehat{q}(t+1)$ during the given prediction time between t_p and $t_p + \varepsilon_r$.

The first step consists of three substeps: (1) initialization, (2) searching and building k_p-NN, and (3) predetermining the potential k_o-NN. In substep (2), various techniques from a linear to an advanced search can be employed to discern object candidates (i.e., $x_j(\tau)$ and o_j) from the historical database, which is beyond the scope of this research. Substep (3) plays a role in predetermining the potential k_o-NN based on the selected k_p-NN, which also reduces the execution time used to determine k_o-NN in the second step.

The second step is composed of four substeps: (1) updating $x_c(t)$, (2) recalculating the state distance, (3) selecting k_o-NN from k_p-NN, and (4) prediction generation. The additional information is updated through substeps (1)-(2), after which (promising) k_o-NN instances are determined in substep (3). These processes are carried out almost instantly without any computational work for accessing and searching through vast amounts of historical data. Finally, the prediction generation is conducted based on the selected k_o-NN using FF.

4. Evaluation and Findings

4.1. Design of Experiments. The efficacy of the proposed KNN-S2S algorithm was examined through an experimental study that had three aspects: the features of the test data, the selection of benchmark models, and the selection of performance measures.

The one-year traffic volume data used in this study were collected from a toll collection system with a five-minute aggregation in 2016. The test bed was a tollgate near Seoul, South Korea, located on motorway #1 (Figure 2(a)). The data array size was 105,408 measurements (288 intervals per day for one year = 288 × 366). The target day for the prediction simulation was the last day of the year (a Saturday); thus, the states of the target data exhibit different temporal developments compared to those of the six previous Saturdays (Figure 2(b)). Noticeably, the lower and upper states during two time periods (05:00–12:00 and 12:00–24:00) deviate from the corresponding recurrent patterns. Due to these nonrecurrent conditions, the target state can be the worst case for KNN-based methods, the prediction accuracy of which depends strongly on the selection of similar patterns.

If the data segment is correct, the accuracy performances of a KNN model either using the data segment or using entire data should be at least comparable. To examine this, the SKNN algorithm widely applied in related works [1–5] is selected as a basic model. The SKNN algorithm, which consists of three steps (initialization, the search for and the building of k-NN, and forecast generation), is executed at t_p. In our study, steps (1) and (2) of SKNN are identical to substeps (1) and (2) of KNN-S2S step (1), respectively, with $x_c(t)$, $l = 0$, and $k = k_o$. Then, the future estimation is generated with the FF defined as (6). A linear search technique that checks each case of a list sequentially until all cases have been searched was used to find the best k-NN in this study. In this way, the performance outcomes of SKNN with the best k-NN information can provide a baseline for the evaluation of the potentialities of KNN-S2S. Note that substep (3) of KNN-S2S step (1) for k_o objects was added to SKNN and KNN-S2S before the forecast generation step for the respective case for analysis purposes, although it is not necessary in actual use. Unfortunately, a dynamic segmentation method similar to the method proposed in this study has not been reported in ITS forecasting research. Therefore, three static segmentation methods (i.e., fixed-window (KNN-D) and flexible segmentation (KNN-U) methods and a combination of KNN-D and U (KNN-UD)) were selected to demonstrate the robustness of the proposed method. Note that advanced readers can refer to the three compared methods in [5].

Based on the features of the target system and the analytical necessities of the five models, the following five performance measures were carefully chosen. The mean absolute percentage error (MAPE) and the root-mean-squared error (RMSE) defined in (7) and (8), respectively, were used for an analysis of the prediction accuracy. The relative percentage error (RPE) was selected as two types, the mean (MPRE) and standard deviation (SDRPE) of RPE [$= (\widehat{q}_i - q_i)/q_i \times 100.0$], to check the conformity of predictions between entire data and segmented data. To examine the searching speed, data usage [= used data/entire data × 100.0] was also used as soft computing.

$$\text{MAPE}(\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\widehat{q}_i - q_i\right|}{q_i} \times 100, \quad q_i > 0 \tag{7}$$

$$\text{RMSE}(\text{veh}) = \left(\frac{1}{N}\sum_{i=1}^{N}\left|\widehat{q}_i - q_i\right|^2\right)^{0.5}. \tag{8}$$

Here, N is the number of test samples and q_i and $\widehat{q}_i$ represent the actual volume and the predicted volume of sample i, respectively, where $i \in n$.

Given the $x_c(t)$ without $q(t)$, k_o and k_p values, d value:
(where $1 < k_o < k_p$; $k_p/n \to 0.0$)
Step 1: Build k_p-NN
If (the previous $t_p + \varepsilon_r$) $< \tau_{nw} < t_p$ then l-value = 1
(where τ_{nw} is the present running time; ε_r is a run-time limit)
(1) Initialize the list of the k_p-objects shown as (5).
(2) For each $x_j(\tau)$ and o_j in the past database
(where $j = 1, 2, \ldots, n$; $o_j = [q_j(\tau + 1), u_j(l, d)]$)
(2-1) Calculate $u_j(l, d)$ between $x_c(t)$ and $x_j(\tau)$ by eq. (4)
(2-2) If $u_j(l, d) < u_{\max}$ then
(where $u_{\max} = \max\{u_1(l, d), u_2(l, d), \ldots, u_{k_p}(l, d)\}$)
(2-2-1) Delete $x_i(\tau)$ and o_i from the k_p-object list
(where o_i is associated with $u_{\max}$, $1 \le i \le k_p$)
(2-2-2) Update $x_j(\tau)$ and o_j into the k_p-object list
(2-2-3) Search $u_{\max}$ in the updated k_p-object list
(3) Sort the select k_p-objects ascending based on $u_i(l, d)$
Step 2: Decide k_o-NN and estimate $\widehat{q}(t + 1)$
If $t_p \le \tau_{nw} \le (t_p + \varepsilon_r)$ then
If $q(t)$ is available then l-value = 0
(1) Add $q(t)$ to $x_c(t)$
(2) Recalculate all $u_j(l, d)$ in the k_p-object list as follows
(where $j = 1, 2, \ldots, k_p$)

$$u_j(l, d) = \left[\left\{u_j(1, d)\right\}^2 + \left\{q(t) - q_j(\tau)\right\}^2\right]^{1/2}$$

(3) For each $x_j(\tau)$ and o_j in the k_p- object list
(where $j = k_o + 1, k_o + 2, \ldots, k_p$)
If $u_j(l, d) < u_{\max}$ then
(where $u_{\max} = \max\{u_1(l, d), u_2(l, d), \ldots, u_{k_o}(l, d)\}$)
(3-1) Delete $x_i(\tau)$ and o_i from the k_o-object list
(where o_i is associated with $u_{\max}$, $1 \le i \le k_o$)
(3-2) Update $x_j(\tau)$ and o_j into the k_o- object list
(3-3) Search $u_{\max}$ in the updated k_o-object list
Else l-value = 1
(4) Estimate $\widehat{q}(t + 1)$ by eq. (6)

Algorithm 1: Pseudocode for the KNN-S2S prediction algorithm.

4.2. Parameter Analysis and Findings. The efficacy of SKNN is achieved through pattern recognition, which is wholly reliant on both the d value of state space and the k value of the nearest neighbours. With regard to the d value (i.e., the embedding size) to construct the temporal properties of $x_c(t)$, Takens' definition of $d \ge 2D+1$ with a given D-dimension state space value [1–3] is widely referenced as a minimal embedding size. Thus far, determining a suitable d value mainly depends on the properties of the temporal states of the target system. With regard to the k value, the theoretical condition of KNN (i.e., $n \to \infty, k \to \infty$ with $k/n \to 0$) is restricted in actuality. Past cases (n) included in the available historical data are limited, and a suitable k value should therefore be finite to meet the condition of $k/n \to 0$. The two parameters are also closely related to each other. In this context, the best or optimal values of the two parameters to ensure reliable estimations should be concurrently analysed [1, 3] and determined in advance. As such, a prediction simulation [1–3, 5, 10, 12, 13] was also used in this work in order to determine suitable parameter values for the two algorithms (SKNN and KNN-S2S).

The SKNN algorithm was experimentally executed to estimate the target data with all combinations of possible parameter values (1,000 cases = d-values [1–10] × k-values [1–100]), and the prediction error of each case was analysed with MAPE. The effects of the two parameters on the prediction errors are shown in Figure 3(a). The best d and k values, $d_o = 5$ and $k_o = 41$, were determined under the condition in which MAPE is minimized to 6.172.

With regard to the d value, the overall MAPEs decrease to the minimal error space and then increase as the d value increases, and the k value exceeds 30. The values of $d_o \pm 2$ are acceptable as optimal d values within the minimal error rate of +0.15%. Specifically, the optimal d values to guarantee reliable estimations satisfy Takens' definition (i.e., a minimum threshold), whereas a maximal threshold also exists. This fact indicates that the temporal state evolution of the optimal d-size is closely related to the pattern recognition process of KNN for future states in some way, and therefore this state of the optimal $d - 1$ size can, at least, play a role comparable to that of the optimal d-size with an acceptable margin of error.

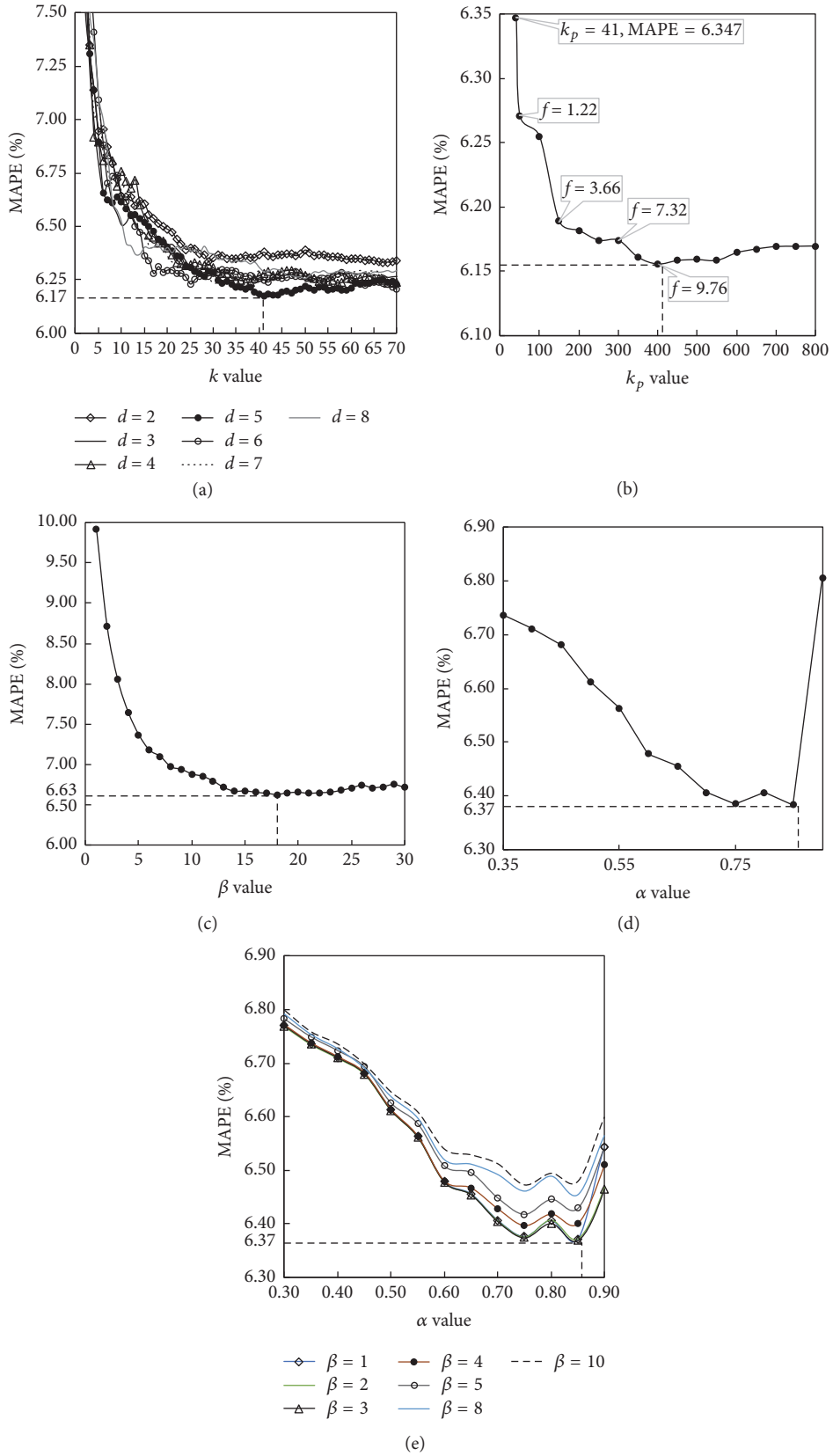


FIGURE 3: Effects of model parameters on prediction errors. (a) Effects of d and k values; (b) effects of the k_p value with d_o and k_o values; (c) effects of the β value; (d) effects of the α value; and (e) effects of α and β values.

With regard to the k value when $d_o = 5$, MAPE decreases exponentially to the minimum point at $k_o = 41$ and then gradually increases with little variation as the k value increases. In this way, the k_o value of 41 satisfies $k/n \to 0$ with 41/105,120 = 0.00039. This essentially implies that a clustered pattern exists in past cases regardless of whether the boundaries between patterns are explicit. The steep decrement of MAPE also provides evidence that the convergence process quickly reaches the geometric centroid of past future states associated with the selected pattern. Additionally, it was noted that suitable values of the two parameters can easily be recalibrated and updated on a periodic time basis (daily, weekly, or even monthly) in advance [1–3, 10]. Furthermore, the d_o value can be fixed after it has been determined through experimental tests [1, 3].

The KNN-S2S algorithm was also conducted with the identified d_o and k_o values, and its performance according to the k_p values ranging from 50 to 800 in increments of 50 is shown in Figure 3(b). When $k_p = k_o$, the prediction error of KNN-S2S guarantees the best case of SKNN within +0.2%. This directly indicates that $x_c(t)$ with $l = 1$ is closely linked to future states and can therefore play an effective role in the selection of k_p-NN, which includes the best k-NN according to $x_c(t)$ with $l = 0$. The MAPEs decrease steeply ($k_p = 41 \to 150$), then gradually decrease ($k_p = 150 \to 400$) to the SKNN performance baseline level of −0.02, and then gradually increase to the baseline of −0.002, as the k_p value increases. In this manner, the performances of SKNN and KNN-S2S are at least comparable in terms of the prediction accuracy when $1.22 \leq f$, whereas KNN-S2S clearly outperforms SKNN when $7.32 < f$, where $f = k_p/k_o$. In the case of $7.32 < f$, it can be seen that the double decision-making process of KNN-S2S is more reliable than the one-time decision of SKNN, as few NNs that are included in the k_o-NN of SKNN according to the rank of $u(0, d)$ with $|q(t) - q(\tau)|^2 \to 0.0$ are excluded from the k_p-NN by the rank of $u(1, d)$.

With regard to a suitable k_p value, if the k_p value is sufficiently larger than the k_o value, reliable prediction accuracy is at least guaranteed by approximating the best case of SKNN within an acceptable error margin, which is insensitive to the prediction accuracy. In this case, the insensitivity is also one of the strong advantages gained during actual use. In addition, the optimal value of k_p can be analysed and updated on a weekly or monthly basis in advance or even fixed at $k_p = f \times k_o$ with $f > 1.0$, through experimental tests.

The three benchmark segmentation methods (i.e., KNN-D, KNN-U, and KNN-UD) were also experimentally evaluated to predict the target data, and all of the past data associated with the previous same day type was used. As for KNN-D, prediction errors decrease exponentially and gradually to MAPE 6.63, when the β value increases (Figure 3(c)). In addition, the prediction error of KNN-D is exactly the same as that of SKNN with $\beta \to 0.5 \times S_n$. This fact indicates that the time dependence exists to a certain extent, even though it is not high. With regard to KNN-U, average prediction error decreases to an optimal error space, goes through the error space area, and then increases steeply according to the increment of the α value (Figure 3(d)). This fact implies that if a high α value is used to reduce searching time, undesirable prediction results (that are not comparable to those of KNN-D) can occur inevitably. In spite of this, KNN-U surely outperforms KNN-D in terms of MAPE, when the span of the α value is between 0.5 and 0.85. In the case of KNN-UD, a distinguished improvement of prediction accuracy does not show, even when the β value increases (Figure 3(e)). This is due to the fact that the temporal variation of the target day deviates from the recurrent pattern of the same day type. This fact directly reveals that a flexible segmentation method combined with a fixed-window method can fail in the improvement of prediction accuracy and can reduce the accuracy performance of KNN-U at least in the case of a nonrecurrent pattern. In addition, KNN-UD is more reliable than KNN-U, when α values are greater than 0.85.

Based on the results of a parameter analysis, the optimal values of the model parameters for SKNN and KNN-S2S were determined to be $[d_o = 5, k_o = 41, k_p^o = 400]$ for result analysis. The optimal values of the model parameters were also identified as $\beta_o = 18$, $\alpha_o = 0.85$, and $[\beta_o = 3, \alpha_o = 0.85]$ for KNN-D, KNN-U, and KNN-UD, respectively, as shown in Figure 1.

4.3. Results and Findings. The test results are summarized in Table 1. The two best performers were KNN-S2S and SKNN, which are followed by the two second-best performers, KNN-UD and KNN-U. This indirectly indicates that the S2S search method, a sort of a dynamic targeting search with an incomplete current time series of traffic flow states, is more reliable in building k-nearest past observations than predetermined static segmentation methods. KNN-S2S is very similar to SKNN in terms of four performance measures, except for data usage. This desirable result directly indicates that the prediction reliability of the high-speed framework proposed in this paper is at least comparable to that of a KNN method, even though the 1.34% of past data was only used at a prediction time point. On the other hand, KNN-S2S slightly outperforms SKNN in terms of prediction-error measures. These results imply that the select pattern built through the pattern recognition process of SKNN, which includes the biased contribution of $|q(t) - q(\tau)|^2 \to 0.0$, could not be the best case for the decision of future states, despite the fact that it would be the best case for the reconstruction of the current temporal state evolution. Therefore, it can be seen at least in our case that the pattern-selection capability of KNN-S2S with the two-step search strategy based on the intrinsic relationship of temporal state evolution is more reliable than that of SKNN from the perspective of the decision-making process.

The conformity of predictions between SKNN using a whole dataset and a KNN model combined with a segmentation method is very crucial. It should be noted that the proposed high-speed KNN framework is not to improve the prediction accuracy but to extremely reduce the execution time of a KNN model, thus ensuring the prediction accuracy of the KNN method combined with the KNN framework. The conformity of SKNN and the two models (KNN-S2S and KNN-UD) through a point-to-point analysis of the

Table 1: Summary of the analysis results.

Model	Performance measures				
	MAPE	MRPE	SDRPE	RMSE	Data usage (%)
SKNN	6.17	0.36	8.13	25.84	100.00
KNN-S2S	6.16	0.36	8.10	25.79	1.34
	0.07*	0.00*	0.19*		0.38**
KNN-D	6.63	0.57	8.80	28.89	12.85
	3.51*	0.29*	4.72*		
KNN-U	6.37	−0.01	8.43	26.24	22.52
	1.68*	−0.40*	2.36*		
KNN-UD	6.37	−0.01	8.43	26.24	22.61
	1.68*	−0.40*	2.36*		

Note. *Measures between elements of SKNN and that of a compared model, and **data usage in the case of one-year data.

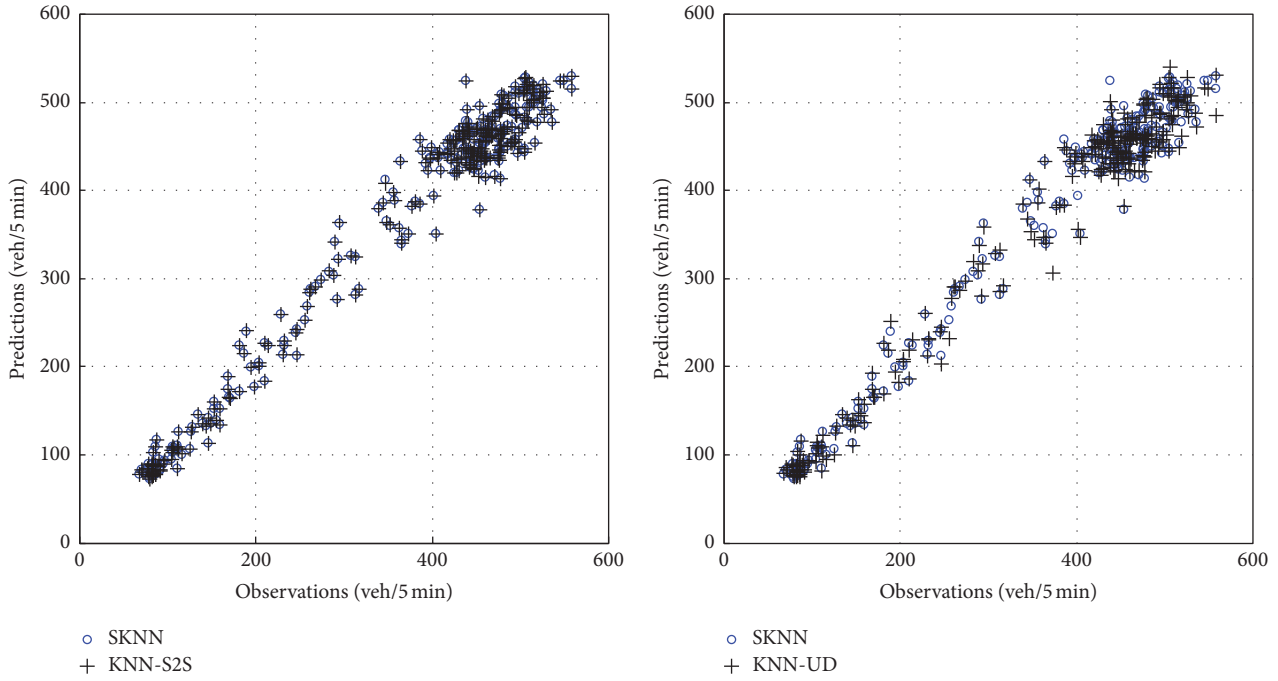


Figure 4: Conformity of predictions between two models.

predictability is shown in Figure 4. The conformity of KNN-S2S is more reliable than that of KNN-UD in terms of MRPE* and SDRPE* (Table 1). The MRPE* and SDRPE* of KNN-S2S are 0.00* and 0.19*, respectively. Otherwise, they are −0.40* and 2.36*. Therefore, it appears that the capability of S2S for dynamic pattern selection is at least comparable to that of the entire data searching and is more reliable than that of a predetermined static segmentation method. This fact provides clear evidence that $x_c(t)$ with $l = 1$ is closely linked to $x_c(t)$ with $l = 0$ in some way from the perspective of the temporal development of traffic volume states.

To inspect the potential of KNN-S2S for selecting promising nearest neighbours, an analysis of the past time points of the selected neighbours was conducted based on time dependency for the target day. The selected past time points (SPTP) by KNN-S2S are compared according to prediction time points (PTP) on the data segment used for the analysis for KNN-UD in Figure 5. Note that SPTP by KNN-UD is included in the data segment. The SPTP (at PTP between 72 and 264) mainly range between 144 and 264. Especially for nonrecurrent cases, SPTP values are highly biased from the space of the data segment. SPTP densely exists between 144 and 264 when PTP ranges from 80 to 144. SPTP also is concentrated in an area between 144 and 228 in the case that PTP ranges from 240 to 264. These results directly indicate that a static data-segmentation method can inevitably fail to ensure the finding of the first- or second-best patterns, even if the data segment might be analysed and updated on a daily or hourly basis with overcoming the time-extensive execution time of the segmentation method.

To examine the potential of KNN-S2S to shorten the execution time, an analysis of data usage (%) according to

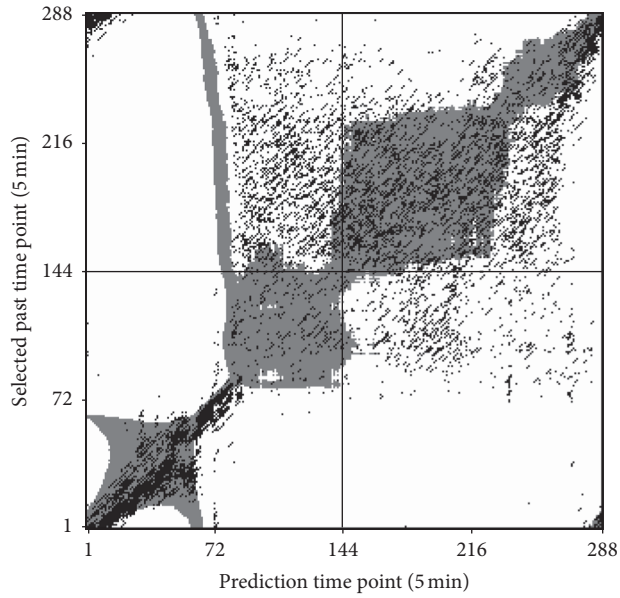


Figure 5: Comparison of a fixed data segment and S2S-based selected time points.

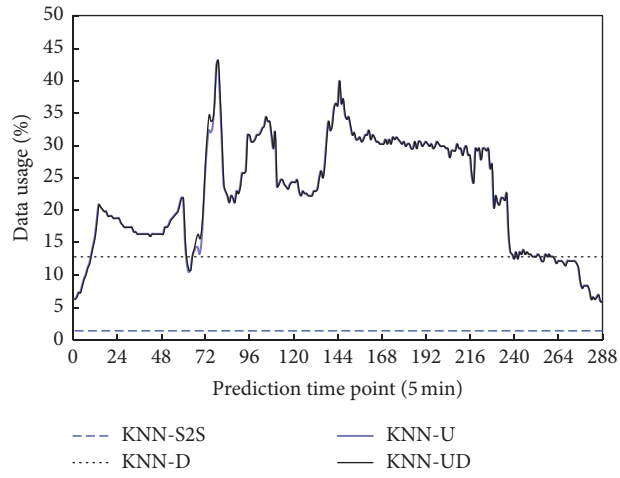


Figure 6: Data usage according to prediction time points.

prediction time points was conducted as shown in Figure 6. This type of analysis is very useful, as the execution speed of the KNN predictor relies considerably on the search time for the amount of historical data. The data usage of KNN-UD varies from 27.78 to 43.06 for the time period (72–240) when estimations generated by a prediction model are heavily used. It should be noted that a volume of searched past data by static segmentation methods increases in proportion to the increment of used past data. For this reason, KNN-UD cannot satisfy the limited running time required in time-critical ITS systems. Contrarily, the data usage of KNN-S2S at t_p is only 1.34% with no computational work related to the access and/or search of historical data at t_p. However, computational loads equal to those of SKNN to search through all of the historical data are inevitable during nonprediction time (i.e., time interval (t)). The computational speed of KNN-S2S was also at least 20 times faster than KNN-UD in the case of daytime. The data usage is also at most 0.38 ($= k_p^o/n \times 100.0 = 400/105{,}120 \times 100.0$) in the case of one-year data, as the useful data is predetermined with the k_p^o value during nonprediction time. This means that the execution time of KNN-S2S at t_p can be 263($= 100.0/0.38$) (i.e., n/k_p^o) times faster than that of SKNN. Therefore, the execution performance of KNN-S2S is instantaneous at t_p. Importantly, the execution time of KNN-S2S does not slow down when the quantity of past data increases, which means that KNN-S2S can meet the required time of ITS data flow consistently. This is one of the strong capabilities of KNN-S2S in real-life applications. Moreover, advanced search methods and data structures can easily be combined into the first step of the KNN-S2S search algorithm to manage the search time more effectively.

5. Conclusions

The slow computing problem of data-driven KNN-NPR remains an ongoing issue as the amount of historical data available grows ever larger. To address this problem, an online framework for a high-speed KNN-S2S algorithm was proposed to reduce the execution time greatly while also improving the prediction accuracy. The algorithm framework was developed based on the deeply linked features of temporal state evolution, sparing artificial approaches to scale down the amount of useful past data and the complexities of advanced search methods.

The analysis results showed that KNN-S2S, despite its extremely high-speed execution time, is at least comparable to the standard KNN method in terms of prediction accuracy. It was also found that the two-step decision-making process of KNN-S2S based on the causal relationship of temporal state development in the pattern recognition process can efficiently diminish the uncertainties of future states from the viewpoint of forecast modelling. This fact can also be indirect evidence that the temporal evolution of traffic volume states is close to the initial deterministic condition, due to the fact that the theoretical foundation of KNN-NPR is based on chaos theory. Specifically, in the case of nonrecurrent conditions in which useful neighbours are not located in a narrow past time window or segment around the prediction time, KNN-S2S is clearly superior to static data-segmentation approaches when used to reduce candidate neighbours.

With regard to the actual feasibility of the KNN-S2S algorithm, it was demonstrated that the algorithm ensures instantaneous and stable execution times with no burden of accessing and retrieving historical data at prediction points, which is in turn another strength over the synchronization and communication of information without any delay in the data flow of time-critical systems. The algorithm was also designed to handle considerable amounts of historical data efficiently without the support of cutting-edge search engines and/or data structures. For this reason, the algorithm can be instantly installed and employed in conventional ITS systems without raising budget issues.

To achieve the objective of high-speed performance, the proposed framework of KNN-S2S was presented with basic

components which are widely used in KNN-based forecast modelling. There are, hence, still promising opportunities to improve on its prediction performance with a combination of different types of state vectors, similarity measures, and forecast functions. Furthermore, the framework can be also used as a preprocess to determine similar learning cases in advanced ITS forecasting models based on support vector machine or deep learning.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the University of Incheon (International Cooperative) Research Grant in 2013.

References

[1] B. Yoon and H. Chang, "Potentialities of data-driven nonparametric regression in urban signalized traffic flow forecasting," *Journal of Transportation Engineering*, vol. 140, no. 7, pp. 143–158, 2014.

[2] B. L. Smith, B. M. Williams, and R. Keith Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 4, pp. 303–321, 2002.

[3] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *IET Intelligent Transport Systems*, vol. 6, no. 3, pp. 292–305, 2012.

[4] B. L. Smith and R. K. Oswald, "Meeting real-time traffic flow forecasting requirements with imprecise computations," *Computer-Aided Civil and Infrastructure Engineering*, vol. 18, no. 3, pp. 201–213, 2003.

[5] M. Bernas, B. Placzek, P. Porwik, and T. Pamuła, "Segmentation of vehicle detector data for improved k-nearest neighbours-based traffic flow prediction," *IET Intelligent Transport Systems*, vol. 9, no. 3, pp. 264–274, 2015.

[6] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Statistical methods for detecting nonlinearity and non-stationarity in univariate short-term time-series of traffic volume," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 5, pp. 351–367, 2006.

[7] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: overview of objectives and methods," *Transport Reviews*, vol. 24, no. 5, pp. 533–557, 2004.

[8] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, no. 1, pp. 3–19, 2014.

[9] B. Yu, X. Song, F. Guan, Z. Yang, and B. Yao, "k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition," *Journal of Transportation Engineering*, vol. 142, no. 6, Article ID 04016018, 2016.

[10] H. Chang, D. Park, S. Lee, H. Lee, and S. Baek, "Dynamic multi-interval bus travel time prediction using bus transit data," *Transportmetrica*, vol. 6, no. 1, pp. 19–38, 2010.

[11] S. Clark, "Traffic prediction using multivariate nonparametric regression," *Journal of Transportation Engineering*, vol. 129, no. 2, pp. 161–168, 2003.

[12] H. Chang, D. Park, Y. Lee, and B. Yoon, "Multiple time period imputation technique for multiple missing traffic variables: Nonparametric regression approach," *Canadian Journal of Civil Engineering*, vol. 39, no. 4, pp. 448–459, 2012.

[13] R. E. Turochy, "Enhancing short-term traffic forecasting with traffic condition information," *Journal of Transportation Engineering*, vol. 132, no. 6, pp. 469–474, 2006.

[14] A. Stathopoulos and M. Karlaftis, "Temporal and spatial variations of real-time traffic data in urban areas," *Transportation Research Record*, no. 1768, pp. 135–140, 2001.

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 7905140, 10 pages
https://doi.org/10.1155/2018/7905140

Research Article

Trajectory Specification Language for Air Traffic Control

Russell A. Paielli

NASA Ames Research Center, Moffett Field, CA 94035, USA

Correspondence should be addressed to Russell A. Paielli; russ.paielli@nasa.gov

Received 10 August 2017; Revised 8 January 2018; Accepted 21 January 2018; Published 7 March 2018

Academic Editor: Rodrigo Carlson

Trajectory Specification is a method of specifying aircraft trajectories with tolerances such that the position at any given time in flight is constrained to a precisely defined bounding space. The bounding space is defined by tolerances relative to a reference trajectory that specifies position as a function of time. The tolerances are dynamic and are based on the aircraft navigation capabilities and the traffic situation. This paper proposes a standard Trajectory Specification Language (TSL) based on the Extensible Markup Language (XML) to represent these specifications and to communicate them by datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The XML format can be converted to binary for operational use, if necessary, using Efficient XML Interchange (EXI) or Abstract Syntax Notation (ASN.1).

1. Introduction

Air traffic control is currently performed by human controllers using radar displays of traffic and voice communication with pilots. The number of flights that a controller can reliably manage at one time, however, is substantially less than the number that could safely fly in the airspace with an automated ATC system [1, 2]. Controllers are remarkably reliable overall, but they are human and therefore make mistakes. Over 1,800 operational errors (breaches of minimum required separation officially attributed to controller error) occurred in one recent year in the US, including 55 serious cases in which "a collision was barely avoided" [3]. Automation can reduce human error, but an autonomous ATC system that works for all possible traffic situations and conditions is difficult to design and implement and is even more difficult to verify and validate to the required level of reliability and integrity.

Trajectory Specification is a proposed far-term enhancement of the Advanced Airspace Concept (AAC) being developed by NASA for automating ATC in both enroute airspace [4–6] and the terminal airspace around major airports [7, 8]. The Trajectory Specification concept was first published in 2005 [9] and has been updated in more recent publications [10–12] (and issued a US patent). A similar proposal by others [13] followed several years after the first publication on the concept.

The main idea of Trajectory Specification is to limit the allowed deviation from an assigned reference trajectory so that the aircraft position at any given time in flight is constrained to a precisely defined volume of airspace. As will be explained later in the paper, the bounding volume at any time is defined by tolerances relative to a reference position at that time as the flight advances along its route. Although near-term applications are possible, the full concept is considered "far-term" because it requires new aviation standards and a new generation of airborne Flight Management Systems (FMS).

Trajectory Specification generalizes Required Navigation Performance (RNP) [14, 15] to the longitudinal plane by adding vertical and along-track tolerances to the cross-track tolerances that are already part of RNP. Dynamic RNP [16] allows routes to be created and assigned dynamically, and it also allows discrete altitude constraints and a required time of arrival (RTA), but it does not specify a continuously bounded trajectory.

Among the potential benefits of Trajectory Specification is the mitigation of risks that arise in the case of a system outage. Safety must be maintained even if the ATC system or the datalink goes down for an extended period of time while traffic density is too high for a human controller to safely take over and manage the traffic. One alternative is to stop any new traffic from entering the affected airspace when

its ATC system or datalink fails [17]. The current traffic will then exit the affected airspace (or land as planned) within approximately 10 to 15 minutes based on the deconflicted trajectories assigned by AAC. The deviation from those trajectories is not bounded; however, conflicts could still arise due to inaccuracies in the winds, weight, or thrust levels that were used to predict the trajectories. The problem could be mitigated by adding an extra separation buffer to the assigned trajectories, but that would diminish airspace capacity during normal operation.

By explicitly bounding deviation from the assigned trajectory in all three axes, Trajectory Specification can guarantee safe separation between flights for as long as they remain in conformance with the tolerances of their assigned trajectories, out to the conflict-free time horizon that was computed. That conflict-free time horizon would normally be on the order of 15 to 30 minutes or more, depending mainly on the current wind modeling accuracy. If the ATC system or datalink goes down, the previously deconflicted and assigned trajectories will remain active in the FMSs to keep the flights safely spaced and separated.

As a fundamentally proactive rather than reactive approach to ATC, Trajectory Specification can also provide safety benefits during normal operation. Rather than simply relying on continuous conflict detection and tactical maneuvering when necessary to correct for prediction errors, it facilitates more rigorous, precise, and predictable strategic planning. Tactical backup systems [18, 19] will still be needed, but they should have to intervene less often. The airborne collision avoidance system (ACAS) would also still be maintained as an emergency backup. The added precision and predictability provided by Trajectory Specification could also facilitate closely spaced parallel approaches or, eventually, formation landing of more than two flights in closely spaced formations to increase runway landing rates.

This paper proposes a standard Trajectory Specification Language (TSL) based on the Extensible Markup Language (XML) to represent these specifications and to communicate them by air/ground datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The underlying datalink technology that would be used is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services (ATS). The paper does not formally define an XML schema but rather shows how the format might be structured and what information it should contain.

Researchers at Boeing have developed the Aircraft Intent Data Language (AIDL) [20], which captures the details of how the pilot intends to fly the aircraft. This language provides detailed input for a ground-based trajectory predictor, thereby facilitating more accurate trajectory predictions for ATC. However, trajectory predictions based on AIDL depend on wind modeling, and large wind modeling errors at any particular time can cause large trajectory prediction errors. The FMS will not have the data it needs to compensate for the wind errors and bound the resulting position error. TSL, on the other hand, provides the predicted trajectory to ATC directly from the FMS and provides tolerances for the FMS to stay within bounds that can guarantee safe separation.

TSL is also similar in some respects to the Extended Predicted Profile (EPP), a downlink capability that was recently added to Automatic Dependent Surveillance–Contract (ADS–C). EPP can downlink predicted state and trajectory data for up to 128 TCPs (Trajectory Change Points: changes in target altitude, heading, or speed). However, not all variables that affect the trajectory reconstruction are included in EPP, and hence predictions based on it can still have significant error, particularly during climb and descent [21]. The time between TCPs can be over 20 minutes even during climb and descent. However, even if the TCPs were much closer together in time, that would not improve the accuracy of the underlying predictions, which depend on wind modeling. Like AIDL, EPP can improve ground-based trajectory prediction accuracy, but it does not bound the errors as TSL is designed to do.

TSL overlaps in scope to some extent with the Flight Information Exchange Model (FIXM) [22], a globally standardized flight data exchange model based on XML. FIXM includes some of the same trajectory data that is in TSL, but it does not fully specify a trajectory with continuous tolerances as TSL does. FIXM is designed primarily for coordination between ATC systems and facilities as well as airline operational centers. Although it will eventually be available on the flight deck, it is not designed for real-time operational coordination between ATC and aircraft in flight. Note also that, like FIXM, trajectory assignments based on TSL must be shared with all ATC facilities that will handle a particular flight.

The remainder of the paper is organized as follows. The next section outlines the Trajectory Specification concept for background. Section 3 presents the proposed Trajectory Specification Language. Section 4 briefly discusses the data transfer requirements of the language and compares it with a common consumer data streaming application for entertainment. The paper ends with a brief summary, and an appendix briefly introduces and discusses XML, EXI, and ASN.1.

2. Trajectory Specification Concept

Trajectory Specification is essentially the construction of dynamic, virtual roadways, corridors, or tubes in the sky using data standards, an air/ground datalink, and software to specify the parameters. Because the parameters are a continuous function of time, it is more precise, more continuous, more dynamic, and more flexible than the static published routes and discrete altitude restrictions that are currently used to organize traffic and separate arrival streams from departure streams in terminal airspace.

A route is the vertical projection of a trajectory onto the surface of the earth. In the Trajectory Specification concept, a route consists of alternating straight (i.e., great circle) segments and circular turn arcs. Any point along the route can be specified by the distance along the route relative to an arbitrary reference point on the route (positive in the direction of flight), and that distance will be referred to as

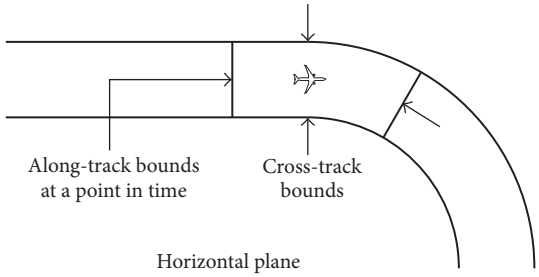


Figure 1: Trajectory bounds in the horizontal plane [12].

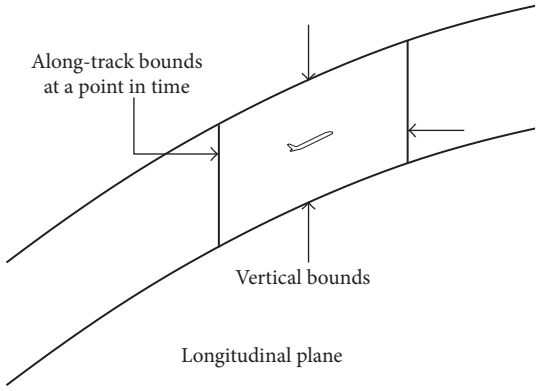


Figure 2: Trajectory bounds in the longitudinal plane [12].

the along-track distance or position. A useful convention for terminal airspace is to define the runway threshold as the zero reference point, so that departure trajectories start at, and arrival trajectories end at, zero along-track distance. Then the along-track position indicates the distance flown from takeoff or yet to be flown to landing.

Figure 1 shows an example of a plan view of trajectory bounds at an instant in time. The lane width is twice the cross-track (lateral) tolerance. The along-track bounds at a point in time are vertical rectangles normal to the route direction (which appear as line segments in the plan view) and are defined by the along-track tolerances relative to the reference position at that time. The along-track bounds combine with the cross-track bounds to form a bounding area in the plan view that is a rectangle in the straight segments, an annular area in the turns, or a combination of the two, as shown in the figure.

Figure 2 shows an example of a side view of trajectory bounds at an instant in time during a climb. The along-track bounds combine with the vertical bounds to form a shape with straight vertical sides and curved top and bottom in the longitudinal plane. In level flight, the top and bottom would also be straight. The vertical tolerances in level flight could be ±100 or ±200 ft, but in climb or descent they could be larger, on the order of ±1000 ft or more, depending on the traffic situation, because altitude is more difficult to predict and control in climb or descent. The tolerances can vary as a function of along-track distance, but the function itself will be fixed at the time of assignment (or reassignment).

The bounding volume at each point in time is a segment or "slice" of a stationary (earth-fixed) bounding tube through which the aircraft is required to fly. Each tube is dynamically constructed for one flight. The vertical cross-sections of the tube normal to the route direction are vertical rectangles, and position along the tube is temporally constrained. (These tubes should not be confused with another tube concept that allows many flights to fly in parallel in a single tube like cars on a freeway.) If one such bounding tube goes over or under another with sufficient vertical separation, then separation is guaranteed as traffic on a freeway is guaranteed to be separated from traffic on a road that goes over or under the freeway. If two such bounding tubes intersect or are separated by less than the minimum allowed separation between flights, the specifications must guarantee separation temporally

Trajectory Specification is an extension of trajectory prediction. Trajectory prediction should normally be done by the FMS, which takes the current flight state, the flight intent, and wind data as inputs and computes a trajectory prediction based on an aircraft performance model. Alternatively, an advanced Electronic Flight Bag (EFB) could potentially be used for this purpose. The intent includes the route, airspeed, cruise altitude, and possibly other parameters. The FMS can also (optionally) receive, record, and take into account the currently assigned trajectories of other flights in the vicinity to avoid conflicts while computing its own trajectory request. Taking other trajectories into account will increase the probability that the requested trajectory will be free of conflicts and approved by ATC without modification. The FMS could use some of the same software components that implement the conflict detection and resolution to be discussed later.

The FMS (or EFB) then downlinks the predicted trajectory to ATC as a request. ATC takes the predicted trajectory as an input and adds default tolerances. It then checks the trajectory for conflicts with the current trajectory assignments of other flights, modifies it to resolve conflicts if necessary, and then uplinks it back to the FMS as the assigned trajectory. If a conflict-free trajectory cannot be found, the flight will be delayed until one can be found (by delaying takeoff time for a departure or putting an arrival into a holding pattern near the terminal airspace boundary). The pilot (or the FMS, if programmed to do so) can request a new or updated trajectory at any time, and the ATC system should approve it if there are no conflicts or constraint violations. The ATC system will generate a new or updated trajectory whenever necessary to resolve a conflict.

The basic operational concept can be summarized as follows:

(1) FMS records trajectory assignments from ATC for other flights (optional).

(2) Pilot enters route and intent data into FMS.

(3) FMS computes a deconflicted trajectory prediction.

(4) FMS downlinks trajectory prediction to ATC as a request.

(5) ATC assigns tolerances and checks for conflicts and constraint violations.

(6) ATC modifies trajectory to resolve conflicts and violations if necessary.

(7) ATC uplinks assigned trajectory with tolerances.

(8) FMS flies assigned trajectory to specified tolerances.

Note that the conflict checks by ground-based ATC are essential for several reasons even if the FMS is programmed to receive, and avoid conflicts with, all trajectory assignments to other flights. Firstly, the FMS could miss a trajectory assignment to another flight if the assigning ATC system is out of radio range at the time of the assignment or the signal is not accurately received for any reason. Secondly, an assignment to another flight could occur while the FMS is computing its own trajectory request, resulting in a potentially dangerous race condition. As an added benefit, the conflict checks on the ground serve as a backup in case of an error in the FMS conflict detection software.

Trajectory tolerances will depend on the aircraft navigation capabilities and the traffic situation. The navigation capabilities determine the lower limit of feasible tolerances, and the traffic situation determines the upper limit. The FMS will know its own minimum tolerances but has no inherent incentive to keep its requested tolerances to a minimum, which is why the tolerances should be provided by ATC from an established database of minimum tolerances for each aircraft or aircraft type.

In general, vertical and along-track tolerances would be made as large as reasonably possible while guaranteeing safe separation. The tolerances could be completely disabled or made arbitrarily large when they are unnecessary. The thrust and airspeed adjustments that are necessary to maintain conformance should be relatively small except in rare cases when the wind model that was used to generate the reference trajectory was grossly in error. Periodic updates can adjust for the accumulated effects of wind errors. If the tailwind is stronger than predicted, for example, the reference trajectory can be shifted in time periodically to recenter the flight, but only if the shift causes no conflict and violates no time constraint.

Trajectories in terminal airspace would normally be assigned shortly before entry into the airspace, perhaps 1 or 2 minutes before entry for arrivals and perhaps 30 seconds before the start of takeoff roll for departures. In either case, a tentative trajectory could be computed and assigned earlier and modified at final assignment time, if necessary, to avoid conflicts. Once assigned, trajectories should normally not need to be modified for the entire 10 to 15 minutes that is typically spent in terminal airspace.

The full Trajectory Specification concept requires both a ground-based ATC component and an airborne FMS component. The focus of this paper is the ATC component, for which prototype software has been developed using functional programming methods with the Scala programming language. The resulting software is intended to form the basis for a sound software architecture as well as a starting point for actual operational implementation. The airborne component is outside the scope of this paper, but a brief overview will put it into perspective.

The function of the airborne component is to understand the Trajectory Specifications and keep the flight in conformance with its assigned trajectory to within the specified tolerances. Due to safety criticality and extreme certification requirements, the airborne component will require a major development effort, but the basic ideas are not complicated. The existing flight-control feedback loop can have a lower-bandwidth outer loop wrapped around it to keep the flight within its bounds. It will monitor for proximity (and predicted proximity) to the bounds and make adjustments as necessary. Vertical speed will be adjusted to maintain vertical conformance, and airspeed will be adjusted to maintain along-track conformance. These adjustments will be similar to what a pilot could do through the Mode Control Panel (MCP), but they will be automated.

3. Trajectory Specification Language

The Trajectory Specification concept will require a standard language to represent and communicate the specifications between aircraft and the ATC system. A proposed language called the Trajectory Specification Language (TSL) has been developed based on XML and will be presented in this section. TSL can be used to downlink trajectory requests and to uplink trajectory assignments, and a conforming FMS will be programmed by the manufacturer to understand the language and to keep the flight in conformance. The objective of this paper is not to formally define an XML schema but rather to show how the format should be structured and what information it should contain. Example XML elements will be presented and explained to provide a high-level design that can be used to develop a formal schema.

A language based on XML was proposed in the original paper on Trajectory Specification [9] in 2005. At that time, Trajectory Specification was a concept with no prototype implementation, but now a software prototype has been developed, and the proposed language has evolved substantially. A software prototype has been developed for the ground-based ATC component of the Trajectory Specification concept using functional programming style in the Scala programming language. The XML elements to be discussed below are serializations of the main classes in that software.

As a serialization of the software classes, the TSL has also been useful for software development. Fast-time simulation testing on a full day of traffic can take over an hour to run, and if a conflict is not resolved properly, the TSL can be used to capture the detailed traffic state at the time of the conflict. That capability allows the developer to restart the simulation at the desired state rather than having to wait up to an hour for the same traffic situation to develop in order to test each software change. This usage does not require a DTD or schema.

The design of TSL involves many choices that reasonable people can disagree on, particularly involving the degree of structure. Should each coordinate of a position be a subelement, for example, or should the coordinates simply be concatenated together into a comma-separated list of numbers? XML purists will argue that more structure is better because it reduces the chance of an error. For most XML applications that may be true, but it is not necessarily true

Table 1: Physical units.

Quantity	Unit
Time	Second (sec)
Horizontal length and position	Nautical mile (nmi)
Vertical length and altitude	Foot (ft)
Angle	Degree (deg)

for this safety-critical application because the barrier to entry is very high. FMS certification is a rigorous process with extensive testing, and errors involving the order of coordinate could never survive it. Hence, for simplicity, the approach taken in this paper is to avoid some low-level structure, but that can easily be changed if a future standards committee decides to do so.

Standard aviation units are used as shown in Table 1. Time is given in seconds (sec), horizontal distance or length is in nautical miles (nmi), vertical length or altitude is in feet (ft), and angles are in degrees (deg). Allowing alternate units can enhance flexibility but also increases the risk of error; whether or not to allow them will be deferred to a future standards committee. If alternate units are not allowed, the unit attributes shown in the examples to follow will not be needed. If alternate units are allowed, they should be from a very small set of options to minimize the risk of error because all users need to be able to recognize and convert them.

3.1. Trajectory. The specification for a trajectory is represented by the `traj` element, the structure of which is shown in Box 1. The `name` attribute could be the call sign or any other unique and appropriate identifier. The `time` attribute is the assignment or request time in seconds (unix time). The optional `assign` attribute is a Boolean that defaults to false and is true if the trajectory is an assignment (as opposed to a request or a resolution maneuver candidate). A possible alternative to the Boolean `assign` attribute could be a `status` attribute that is limited to a small set of enumerated values, such as `ASSIGN`, `REQUEST`, and `CANDIDATE`. The `GUFI` attribute holds the Globally Unique Flight Identifier, which is required for all ATC flight data transactions.

The subelements of the `traj` element are `flight` (flight information), `route`, `refTraj` (reference trajectory), `altTols` (altitude tolerances), and `alongTols` (along-track tolerances). As explained earlier, these elements correspond to the Scala classes that were developed to implement the concept. Although not shown, each subelement could have its own optional `name` attribute for reference (which should all be the same).

The `flight` subelement contains basic flight information in its `info` attribute, including the aircraft type code (B738 in this example), the weight class (Large), the airport code (DFW), the runway (18R), and the flight type (Arr for arrival or Dep for departure). This information could be broken down into individual subelements as shown in Box 2, which is more explicit and can be parsed using standard XML tools. However, the simpler form reads naturally and is trivial to parse. A standard equipment code could be appended to the aircraft type code if necessary (after a slash, as usual). The other subelements of the trajectory (`traj`) element are explained in the following subsections.

3.2. Route. The route is the plan view of the trajectory, consisting of straight segments and turn segments. The structure of the `route` element is shown in Box 3. The basic flight information (`flight`) from the `traj` element is repeated for reference because the route depends on that information. The reference along-track starting distance (`startDist`) is arbitrary and was set to −53.031923 nmi to make the along-track distance zero at the end of the trajectory (at the runway threshold). The cross-track tolerance (`crossTol`) is set to 0.6 nmi, which is equivalent to RNP 0.3.

The waypoint list (`waypts`) of the `route` element has a `type` attribute, which can be `local` or `global`, depending on whether the waypoint locations are given in terms of a locally level map projection or global geodetic coordinates (latitude and longitude). If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is D10 in this case, a predefined local frame for the D10 TRACON. A predefined frame (defined by its projection type and tangency point) would be published for each major TRACON and would never change. If a global frame is used, it would default to WGS-84 or whatever a standards committee deems appropriate. As explained earlier, the unit attributes could be optional and default to `nmi` if not given.

The cross-track tolerance should rarely change during a flight except possibly when entering or exiting terminal airspace. The `crossTol` subelement of the `route` element could represent such a change by appending data to the first cross-track tolerance as shown in Box 4. In this example, the initial cross-track tolerance of 0.6 nmi is followed after a slash by `23.5: 2.0`, which means that the cross-track tolerance changes to 2.0 nmi at the along-track distance of 23.574 nmi. Changes in cross-track tolerances are instantaneous (discontinuous) at the change point (rather than piecewise linear as for the vertical and along-track tolerances to be discussed later). Additional changes further along the route could follow similarly if necessary. Additional structure could be added here but, as explained earlier, the FMS certification process is so rigorous that basic parsing errors would not survive it.

The individual waypoints are each specified by the `waypt` element. Each waypoint can have an optional `name` attribute, which is provided for reference only and has no effect on the resulting route. The first waypoint in the example is called "`HOWDY`," and the coordinates of its position are 27.917174 nmi and −31.050092 nmi. If geodetic coordinates are used, the coordinates would be latitude and longitude in degrees. The `rad` subelement represents the turn radius at the waypoint, which is 2.0 nmi for the first waypoint in this case. The turn radius is required for all but the first waypoint for departures or the last waypoint for arrivals (any value provided for those endpoints will simply be ignored as meaningless). All turns are tangent arc "flyby" turns, and if a specified turn radius cannot be accommodated for the given the leg lengths, the route will be rejected as geometrically invalid. The use of six digits after the decimal place provides a high resolution that is

```
<traj name="AAL123" time="1378478953" assign="true" GUFI="...">
    <flight name="AAL123" info="B738 Large DFW 18R Arr"/>
    <route> ... </route>
    <refTraj>...</refTraj>
    <altTols>... </altTols>
    <alongTols>...</alongTols>
    </traj>
```

Box 1: Trajectory structure (XML sample).

```
<flight name="AAL123">
    <aircraft>B738</aircraft>
    <weightClass>Large</weightClass>
    <airport>DFW</airport>
    <runway>18R</runway>
    <type>Arrival</type>
    </flight>
```

Box 2: Flight information (XML sample).

nearly equivalent to binary. More digits would be needed for geodetic coordinates to achieve the same level of resolution. The ellipses indicate that several waypoints have been cut for brevity in this example.

3.3. Reference Trajectory. The reference trajectory is the ideal predicted trajectory that would be flown in the absence of any wind modeling errors or other modeling errors. It is represented by the `refTraj` element, the structure of which is shown in Box 5. The time step (`dt`) in this example is 5.0 sec, and the reference time (`refTime`) is 1378478942.2 sec. The reference time is simply the starting time of the reference trajectory and is used to save space and enhance readability.

Like the `waypts` subelement of the `route` element discussed above, the points list (`points`) has a `type` attribute, which can be `local` or `global`, depending on whether the points are given in terms of a locally level map projection or global geodetic coordinates. If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is again `D10` in this case. The `units` attributes are shown as `"sec,nmi,ft"`, meaning that the time is given in seconds, horizontal position is given in nautical miles, and altitude is given in feet.

Each point (`pt`) in the list of points has a time stamp (relative to the reference time), x and y coordinates of position, and an altitude, all separated by commas and (optional) spaces. The first point has a time of 0.0 sec, x and y coordinates of 27.905328 nmi and −30.981547 nmi, and an altitude of 9986 ft. Again, additional subelement structure could be added here, but the FMS certification process is so rigorous that basic parsing errors would not survive it. As before, the use of six digits after the decimal place provides a high resolution almost equivalent to binary. Again, if geodetic coordinates are used, the horizontal coordinates would be latitude and longitude in degrees, and more digits will be required for the same level of resolution. The ellipses indicate that many points have been cut for brevity in this example.

An alternate form of the point (`pt`) element is also possible. Rather than specifying position in terms of locally level or geodetic coordinates, it could be specified in terms of along-track distance. In that case, each point would contain the time stamp, the along-track distance, and the altitude. This alternate form has two advantages. Firstly, it reduces the number of position coordinates from two to one, significantly reducing storage space and transmission bandwidth requirements. Secondly, it eliminates the possibility that the specified point could be off the route (because the cross-track coordinate would always be zero by definition). The disadvantage, perhaps insignificant, is that the actual position on the surface of the earth is not obvious, and the potential for error may be slightly higher.

Note that the points of the reference trajectory need not be uniformly spaced in time as shown in the example. They could be spaced further apart in steady-state flight, for example, but they should be spaced 5 seconds or less in turns and altitude transients. They will be converted internally by the ATC or FMS software to a uniformly spaced sequence of time steps specified by the `dt` element. (Uniform time steps are a key to efficient computation because they allow fast access of trajectory data points as a function of time by array indexing and interpolation.) The reference trajectory typically takes up approximately 80–90% of the overall storage space for the Trajectory Specification.

3.4. Altitude Tolerances. Figure 3 shows an example of a side view of the stationary, rectangular tube in which an aircraft is required to fly. The Trajectory Specification software automatically detects the level segments and applies the default tolerance of ±200 ft in those segments. It also applies the tapered transitions between the level and nonlevel segments and cuts off the altitude overshoots that would otherwise precede or follow a level segment. The tapered transitions are shown at a slope of 2.5 deg but could be varied slightly. The smaller the taper angle is, the more airspace is reserved. The taper angle should be slightly less than the climb or descent angle to allow enough room for a normal leveloff or start of climb or descent.

Note that the end of climb and start of descent are clearly bounded. Lack of such bounds causes significant uncertainty for automated conflict detection, diminishing airspace capacity [23]. Discretionary descents in particular (in which the pilot is given discretion as to when to start descent) have

```
<route name="1366">
    <flight name="1366" info="B738 Large DFW 18R Arr"/>
    <startDist unit="nmi">-53.031923</startDist>
    <crossTol unit="nmi">0.6</crossTol>
    <waypts type="local" frame="D10" unit="nmi">
        <waypt name="HOWDY"> 27.917174, -31.050092 <rad>2.000</rad></waypt>
        <waypt> 27.235636, -27.108529 <rad>3.000</rad></waypt>
        ...
        </waypts>
    </route>
```

Box 3: Route element (XML sample).

```
<crossTol unit="nmi">0.6 / 23.5: 2.0</crossTol>
```

Box 4: Cross-track tolerance element (XML sample).

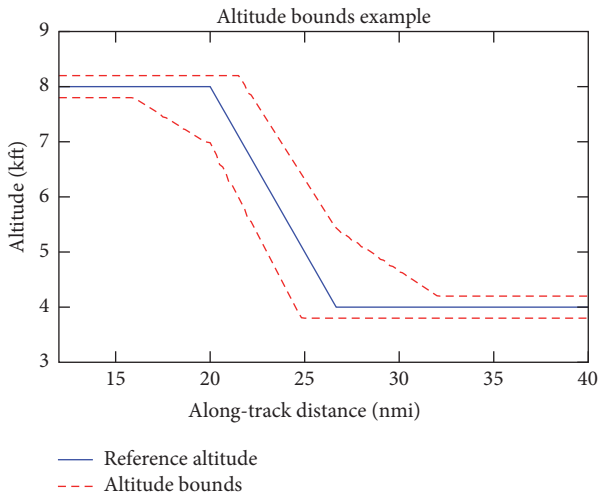


Figure 3: Simplified example of altitude bounds as a function of distance along route [12].

caused problems for automated conflict detection, but they can be accurately represented, if necessary, by using larger altitude tolerances.

The tolerances for level flight are not explicitly specified but are assumed to be consistent with the current "altitude rounding" rule: if an aircraft is in level flight within *pm*200 ft of its cleared altitude, it is considered to be at its cleared altitude for purposes of separation requirements (without this rule, many nuisance alerts would occur due to small altitude deviations in cruise). The taper angle for transition to and from level flight is specified in the optional `taper` subelement, and the default value is 2.5 deg if not specified. The tolerances for level flight (including the tapered transitions to and from level flight) override the explicitly specified tolerances discussed below, which apply in climb and descent.

The altitude tolerances are specified in the `altTols` element, the structure of which is shown in Box 6. The piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement shows vertical tolerances of ±500 ft at an along-track distance of 0 nmi. The lower tolerance is given first as a negative value, followed by the upper tolerance. By definition, those tolerances apply for any along-track distance less than the first specified distance of 0 nmi. The second `tol` point shows that the tolerances increase linearly to ±1000 ft at an along-track distance of 40 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (If there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Although the lower and upper tolerances are equal in this example, in general they can be different.

3.5. Along-Track Tolerances. The along-track tolerances are specified in the `alongTols` element, the structure of which is shown in Box 7. Again, the piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement below shows along-track tolerances of ±1 nmi at an along-track distance of −53.031923 nmi. The back tolerance is given first as a negative value, followed by the front tolerance.

The starting distance was chosen in this example to be the same as the `startDist` of the route, which was set to make the along-track distance zero at the runway threshold. By definition, those tolerances apply for any along-track distance less than that first specified distance. The second `tol` point shows that the tolerances decrease linearly to ±0.2 nmi at an along-track distance of −10 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (Again, if there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Such a decreasing along-track tolerance would be typical during an arrival rush when throughput is critical. Again, although the front and back tolerances are equal in this example, they can be different.

The trajectory tolerances are not required in a downlinked trajectory request from an aircraft because they will

```
<refTraj name="1366">
    <dt unit="sec">5.000</dt>
    <refTime unit="sec">1378478942.200</refTime>
    <points type="local" frame="D10" units="sec,nmi,ft">
        <pt>0.000,27.905328,-30.981547,9986</pt>
        <pt>5.000,27.837196,-30.587082,9982</pt>
        <pt>10.000,27.768812,-30.192529,9981</pt>
        ...
        <pt>750.000,7.024273,-2.354429,481</pt>
        </points>
    </refTraj>
```

Box 5: Reference trajectory element (XML sample).

```
<altTols units="nmi, ft">
    <tol> 0: -500, 500 </tol>
    <tol> 40: -1000, 1000 </tol>
    <taper unit="deg">2.5</taper>
    </altTols>
```

Box 6: Altitude tolerances element (XML sample).

```
<alongTols, unit="nmi">
    <tol> -53.0319: -1.0, 1.0 </tol>
    <tol> -10: -0.2, 0.2 </tol>
    </alongTols>
```

Box 7: Along-track tolerances element (XML sample).

be assigned by ATC based on published aircraft navigational capabilities and the current traffic situation. Allowing the pilot or FMS to specify arbitrary tolerances would not make sense because the only incentive would be to request the largest possible tolerances. If tolerances are allowed in the downlinked trajectory request, they should be the tightest tolerances that the aircraft has been determined to be capable of conforming to. A database should be established so that the ATC system can look up the navigational capabilities of each aircraft based on the aircraft type and equipage code.

3.6. Trajectory Updates. Trajectory updates can often be done more efficiently without sending an entire new trajectory. A simple time shift to adjust for accumulated wind errors, for example, can be specified by the time shift rather than an entire new trajectory. An example of how that could be specified is shown in Box 8. As before, the time attribute represents the assignment (update) time, and the time shift element is the time by which the reference trajectory is shifted, which is −8 sec in this case. When received by the aircraft, the FMS would generate an entire new trajectory representation for its own use, but that trajectory need not take up communication bandwidth.

As another example of a trajectory update, consider changing the tolerances without changing the route or the reference trajectory. That could be done by using the `traj` element discussed earlier and simply omitting the route and the reference trajectory if they are unchanged.

4. Data Transfer Requirements

As mentioned earlier, the underlying datalink technology that would be used for TSL is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services. This new aeronautical datalink technology is expected to significantly increase the bandwidth available for applications such as TSL.

To quantify the data transfer requirements of the proposed language, a typical terminal trajectory was selected at random and serialized using the language. Trajectories through the terminal area typically range from 10 to 15 minutes in length, and the selected trajectory was approximately 15 minutes long. The serialization was found to contain approximately 10.1 kilobytes before compression and 2.6 kilobytes after compression with gzip, for a compression ratio of approximately 3.9. (EXI is better than gzip for compressing XML but was not used here because it is not yet widely available.) The reference trajectory takes up approximately 84% of the storage space.

Enroute trajectories are usually longer than terminal trajectories, up to several hours in length, so the data quantities could be roughly an order of magnitude larger on average. However, enroute trajectories also tend to have more and longer periods of steady-state (straight and level) flight. Steady-state segments can be accurately represented with longer time steps of perhaps 30 sec or more (rather than 5 sec or less as needed for turns and other nonsteady segments), reducing the amount of data required. Note also that enroute trajectories will be updated occasionally to adjust for the accumulated effect of wind errors and to resolve conflicts as they arise within the conflict-free time horizon of approximately 20 to 30 minutes. These updates should be needed perhaps once every 10 to 20 minutes or so, but many of them should be simple time shifts.

To put these data quantities into perspective, consider the amount of data involved in streaming or downloading

```
<traj name="1366" time="1378478953" assign="true" GUFI="...">
    <timeshift unit="sec">-8.000</timeshift>
    </traj>
```

Box 8: Time shifting example (XML sample).

music to a mobile device. A song of 4 minutes in length at a typical resolution of 32 kbps requires roughly 1 MB of data to be transmitted. That is enough for roughly 100 uncompressed terminal area trajectories or 400 such trajectories after compression. Those numbers should be reduced by roughly an order of magnitude for longer enroute trajectories. If these data rates are not acceptable, or if the XML processing is too slow, a binary equivalent of the proposed XML format based on EXI or ASN.1 can be used as discussed earlier.

5. Summary

Trajectory Specification is a proposed far-term enhancement of the Advanced Airspace Concept (AAC) being developed by NASA for automating ATC in both enroute airspace and the terminal airspace around major airports. The main idea is to limit the allowed deviation from an assigned reference trajectory so that the aircraft position at any time instant in flight is constrained to a precisely defined volume of airspace. Trajectory Specification generalizes Required Navigation Performance (RNP) to the longitudinal plane by adding vertical and along-track tolerances to the cross-track tolerances that are already part of RNP.

A Trajectory Specification Language (TSL) based on XML has been developed to serialize Trajectory Specifications and communicate them by datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The proposed language can serve as a starting point for the development of a communication standard for the Trajectory Specification concept. The language is flexible and directly readable by humans, and the bandwidth requirements are modest compared to common consumer data streaming and downloading applications. If necessary, standard compression or efficient binary forms based on EXI or ASN.1 can be used.

Appendix

Extensible Markup Language (XML)

Aeronautical datalink formats have traditionally been binary, but text-based formats such as XML and JSON (JavaScript Object Notation) are viable options for this concept. They are well established standards, and the major programming languages have libraries to parse and process them. Text is more flexible than binary data and is directly readable by humans. Text requires more storage space and bandwidth than binary data, but new technologies are available to convert to an equivalent binary form for more efficient transmission and processing, if necessary. XML is slightly more verbose than JSON, but it was chosen for this safety-critical application because it is a more established standard with more capabilities and better support.

The required structure and form of an XML document can be formally described by a Document Type Definition (DTD) or an XML schema. Alternatively, the equivalent of an XML schema can be created using Abstract Syntax Notation (ASN.1) [24], an industry standard for defining platform-independent binary data formats. ASN.1 includes XML Encoding Rules (XER) to facilitate equivalent binary and XML data formats and conversion between the two. If an ASN.1 schema is developed for the Trajectory Specification concept, both a binary and an XML version of TSL will be available. Another alternative is the Efficient XML Interchange (EXI) [25] format, an efficient binary form of XML.

An XML document consists of a hierarchy of elements, each of which can contain subelements and/or attributes. Consider, for example, the following XML element:

```
<note from="building manager" to="occupants">
    <subject>electrical maintenance</subject>
    <body>Power will be out in building 210 ⋯ </body>
    </note>
```

The main delimiters are angle brackets, which enclose the opening and closing tags of each element or subelement. The example shows an element called `note`, which has attributes `from` and `to`, and which has subelements `subject` and `body`. Attributes are specified in the opening tag of an element, and the closing tag contains the element name preceded by a forward slash. Attribute values should be in quotes as shown. (The convention in this paper is to indent the closing tag one level more than the opening tag to better preserve the logical structure.)

Attributes are supposed to contain metadata, but the distinction between data and metadata is not always obvious. A DTD or schema can restrict allowed values to a specified discrete set. It can also restrict attributes and elements to

specified data types, such as text, Boolean, integer, or decimal number. Other structural and ordering restrictions can also be imposed, but they will not be discussed here.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

References

[1] J. Mercer, J. Homola, C. Cabrall et al., "Human-automation cooperation for separation assurance in future NextGen environments," in *Proceedings of the International Conference on Human-Computer Interaction in Aerospace, HCI-Aero 2014*, Santa Clara, CA, USA, August 2014.

[2] T. Prevot, J. R. Homola, L. H. Martin, J. S. Mercer, and C. D. Cabrall, "Toward Automated Air Traffic Control-Investigating a Fundamental Paradigm Shift in Human/Systems Interaction," *International Journal of Human-Computer Interaction*, vol. 28, no. 2, pp. 77–98, 2012.

[3] J. B. Guzzetti, *FAAΓs Progress and Challenges in Advancing Safety Oversight Initiatives*, US Dept. of Transportation, 2013.

[4] H. Erzberger, "Automated conflict resolution for air traffic control," in *Proceedings of the 25th Congress of the International Council of the Aeronautical Sciences 2006*, pp. 3946-3673, September 2006.

[5] H. Erzberger and R. A. Paielli, "Concept for Next Generation Air Traffic Control System," *Air Traffic Control Quarterly*, vol. 10, no. 4, pp. 355–378, 2002.

[6] H. Erzberger, T. A. Lauderdale, and Y.-C. Chu, "Automated Conflict Resolution, Arrival Management, and Weather Avoidance for Air Traffic Management," *Journal of Aerospace Engineering*, vol. 226, no. 8, 2012.

[7] A. Nikoleris and H. Erzberger, "Autonomous System for Air Traffic Control in Terminal Airspace," in *Proceedings of the 14th AIAA Aviation Technology, Integration, and Operations Conference*, Atlanta, GA, June 2014.

[8] H. Erzberger, T. Nikoleris, R. A. Paielli, and Y.-C. Chu, "Algorithms for control of arrival and departure traffic in terminal airspace," *Journal of Aerospace Engineering*, vol. 230, no. 9, 2016.

[9] R. A. Paielli, "Trajectory Specification for High-Capacity Air Traffic Control," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 361–385, 2005.

[10] R. A. Paielli, "Trajectory Specification for Automation of Terminal Air Traffic Control," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, San Diego, California, USA.

[11] R. A. Paielli, "Trajectory specification for terminal air traffic: Arrival spacing," *Journal of Aerospace Information Systems*, vol. 13, no. 10, pp. 381–392, 2016.

[12] R. A. Paielli, "Trajectory Specification for Terminal Air Traffic: Conflict Detection and Resolution," in *Proceedings of the 17th AIAA Aviation Technology, Integration, and Operations Conference*, Denver, Colorado, USA.

[13] D. M. Finkelsztein, J. L. Sturdy, O. Alaverdi, and J. K. Hochwarth, "4D Dynamic Required Navigation Performance," NASA/CR–2011-217051, 2011.

[14] RTCA DO-283A: "Minimum Operational Performance Standards for Required Navigation Performance for Area Navigation," Oct 2003.

[15] *RTCA DO-236C: "Minimum Aviation System Performance Standards: Required Navigation Performance for Area Navigation," June 2013*.

[16] Federal Aviation Administration: "Dynamic Required Navigation Performance: Preliminary Concept of Operations," Version 1.0, RTCA Paper No. 069-14/PMC-1199, March 2014.

[17] J. W. Andrews, J. D. Welch, and H. Erzberger, "Safety Analysis for Advanced Separation Concepts," *Air Traffic Control Quarterly*, vol. 14, no. 1, pp. 5–24, 2006.

[18] R. A. Paielli, "Evaluation of tactical conflict resolution algorithms for enroute airspace," *Journal of Aircraft*, vol. 48, no. 1, pp. 324–330, 2011.

[19] H. Tang, J. E. Robinson, and D. G. Denery, "Tactical conflict detection in terminal airspace," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 403–413, 2011.

[20] J. López-Leonés, M. A. Vilaplana, E. Gallo, F. A. Navarro, and C. Querejeta, "The aircraft intent description language: A key enabler for air-ground Synchronization in trajectory-based operations," in *Proceedings of the 26th DASC Digital Avionics Systems Conference - 4-Dimensional Trajectory-Based Operaions: Impact on Future Avionics and Systems*, pp. 1–D412, USA, October 2007.

[21] J. Bronsvoort, G. McDonald, M. Paglione et al., "Real-time trajectory predictor calibration through extended projected profile down-link," in *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2015*, June 2015.

[22] "Flight Information Exchange Model Operational Data Description," FIXM version 4.0.0, October 31, 2016. https://www.fixm.aero/.

[23] A. Cone, A. Bowe, and T. Lauderdale, "Robust Conflict Detection and Resolution around Top of Descent," in *Proceedings of the 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conf.*, Indianapolis, In, USA, Sept. 2012.

[24] Abstract Syntax Notation: http://www.itu.int/en/ITU-T/asn1.

[25] Extended XML Interchange: https://www.w3.org/TR/exi/.

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 9317291, 15 pages
https://doi.org/10.1155/2018/9317291

Research Article

Density-Based Statistical Clustering: Enabling Sidefire Ultrasonic Traffic Sensing in Smart Cities

Volker Lücken, Nils Voss, Julien Schreier, Thomas Baag, Michael Gehring, Matthias Raschen, Christian Lanius, Rainer Leupers, and Gerd Ascheid

Institute for Communication Technologies and Embedded Systems (ICE), RWTH Aachen University, Aachen, Germany

Correspondence should be addressed to Volker Lücken; luecken@ice.rwth-aachen.de

Received 18 August 2017; Accepted 6 December 2017; Published 4 January 2018

Academic Editor: Mehmet Yildirimoglu

Traffic routing is a central challenge in the context of urban areas, with a direct impact on personal mobility, traffic congestion, and air pollution. In the last decade, the possibilities for traffic flow control have improved together with the corresponding management systems. However, the lack of real-time traffic flow information with a city-wide coverage is a major limiting factor for an optimum operation. Smart City concepts seek to tackle these challenges in the future by combining sensing, communications, distributed information, and actuation. This paper presents an integrated approach that combines smart street lamps with traffic sensing technology. More specifically, infrastructure-based ultrasonic sensors, which are deployed together with a street light system, are used for multilane traffic participant detection and classification. Application of these sensors in time-varying reflective environments posed an unresolved problem for many ultrasonic sensing solutions in the past and therefore widely limited the dissemination of this technology. We present a solution using an algorithmic approach that combines statistical standardization with clustering techniques from the field of unsupervised learning. By using a multilevel communication concept, centralized and decentralized traffic information fusion is possible. The evaluation is based on results from automotive test track measurements and several European real-world installations.

1. Introduction

In the context of the Internet of Things (IoT), the integral transformation process of urban infrastructure into intelligent and connected devices plays an important role for future mobility. Cities experience an increasingly high level of road congestion due to the focused traffic coming along with progressing urbanization. This congestion induces a high social, economical, and environmental cost. From a European Union (EU) perspective, it is amounted to about one percent of the GDP [1] and predicted to increase by about 50 percent by the year 2050 [2]. From an environmental point of view, carbon dioxide emissions and air pollution, especially locally in cities, are a significant social and health aspect [3]. With this new unprecedented level of traffic density, offline and online traffic planning including live routing techniques becomes a requisite for larger cities, being useful both for end users and for city planners. Especially in the case of real-time traffic monitoring and the subsequent information provision to drivers and public routing systems, proper conditioning of this traffic information in terms of accuracy, area coverage, and density is paramount. Traffic congestion can only be effectively reduced if this information base, which is used for the routing decisions, is delivered with a sufficient level of quality in these dimensions. Then, with a city-wide traffic status, large-scale end-to-end routing optimizations can be performed. The results can be deployed to the vehicle driver with provision methods such as in-car information systems or centralized traffic guidance for triggering the individual's routing reaction. If this provision is not performed properly either due to an insufficient traffic information quality or improper consideration of group effects in the optimization, even detrimental effects can occur [4]. In this article, these requirements on traffic information quality and density are in focus, building a base for future routing decisions. Many currently used traffic sensing techniques for creating this live data basis will only have a negligible positive impact on the routing decisions

if they are distributed sparsely and do not achieve sufficient coverage.

In general, two main types of techniques achieving more comprehensive traffic information have been established in the last years. A first category is the fixed or mobile infrastructural distribution of sensors, like inductive loops and traffic cameras. The second type is represented by systems relying on end-user based distributed information, for example, movement data from smartphone users or information gathered using V2X communication approaches. Details and characteristics of several techniques are discussed in this paper, considering not only metrics like traffic information quality and performance in different scenarios but also possible privacy issues which may arise with certain techniques.

The novel ultrasound-based traffic sensing technique, which is the focus of this paper, belongs to the category of infrastructural sensing. A significant difference to existing ultrasonic sensor systems is the placement of the sensors in a so-called sidefire setup. This means that the sensors are mounted in a height of at least three meters and face the street sideways with downtilted sensors, while being positioned on the side of the street. Previously existing systems were mainly single-lane ultrasonic sensors measuring the height profile top-down, for example, from a special sensor gantry above the street, with a simple first-reflection processing. The monitoring of multiple lanes is possible with our setup, while, at the same time, the requirements in terms of algorithmic evaluation and processing are increased in comparison to these previously existing systems. However, with these improvements, our approach allows operation in a highly reflective acoustic environment, which is present in urban areas and permits simply mounting the sensor on the side of the street. This is enabled by a new approach with a combination of statistical signal processing, clustering, and inference algorithms for traffic participant object detection.

This sidefire ultrasonic traffic sensing technique is part of a development which focuses on intelligent infrastructure solutions, more specifically on intelligent street lights. An exemplary integrated setup is shown in Figure 1. These light modules can be used to replace inner components of street lamps, for example, by retrofitting the widespread type of high-pressure sodium-vapor lamps. Then, they can act as a flexible platform for many applications by employing the processing, communication, and light control actuation capabilities. With the transition to LED-based lighting, which is triggered by high energy costs and several EU directives [5], a large proportion of the existing street lamps will have to be replaced in the next decade, offering potential for an integration of Smart City solutions. The seamless integration into infrastructure allows an almost complete coverage of relevant urban traffic to provide the required coverage density of traffic information. Furthermore, the major problem of power supply for sensing devices is alleviated, as the lamps are connected to the mains power, which eliminates the need for energy-constrained and expensive battery-powered devices.

The rest of this article is organized as follows. Section 2 provides the related work of urban traffic sensing techniques in general and ultrasonic sensing in specific. Section 3 describes the system architecture together with the specific scenario and typical sensor setup. Section 4 provides the algorithmic approach to the sensor signal processing system and object detection inference. In Section 5, the evaluation methodology and framework for reference data are presented. Section 6 gives results of real-world measurements and provides an evaluation from the perspective of detection and classification together with a state-of-the-art technology comparison. Section 7 then concludes the article.

Figure 1: Integrated street lamp head with a single ultrasonic sensor head.

2. Related Work

Ultrasonic traffic participant detection falls into the field of traffic sensing with a large number of techniques available in research and in practical application. In this section, we therefore provide a state-of-the-art overview from two perspectives. First, we give a comprehensive overview of traffic monitoring techniques in general with a discussion of their performance, strengths, and weaknesses in order to show the technological gaps and potential. In the second part, specific technical approaches in ultrasonic traffic sensing, which are rare both in concepts and in practical implementations, are given. This application-centric perspective shows the drawbacks and unresolved problems in current state of the art.

2.1. Traffic Monitoring Techniques from a Broad View. In the introductory section, two main categories of traffic information acquisition systems were introduced. These are on the one hand infrastructure-based sensing systems, working in a temporarily or permanently fixed location. On the other hand, the type of end-user assisted sensing systems incorporates distributed information acquired from the drivers, their technical devices, and the vehicle's internal sensors itself. In the following, a state-of-the-art overview together with a technological comparison is given for nowadays most relevant traffic monitoring techniques of both categories. A condensed overview of the properties is also given in Table 1.

The most widely used approach to traffic sensing is the use of infrastructure-based solutions, which represents a mainly centralized approach. In general, authority of the data is kept with the operator and, thus, several of the problems coming with user-centric techniques do not arise. However, the sensing systems have to be deployed over the whole city or at least at critical locations in terms of traffic. Achieving the required density for an extensive urban traffic monitoring is therefore paramount, which brings along high effort and cost as a centralized solution.

Table 1: Overview of state-of-the-art traffic monitoring techniques.

Sensing technology	Characteristics (⊕/⊖)	References
Infrastructural sensing techniques		
Infrared sensors (passive and active)	⊕ Vehicle class and speed information acquisition is possible ⊕ Multilane operation possible (active sensors) ⊖ Susceptible to bad weather conditions ⊖ Regular cleaning required (active sensors)	[10, 11]
Camera-based systems	⊕ High detection and classification accuracy ⊖ Privacy and authorization issues ⊖ Susceptible to bad weather and light conditions	[6, 7]
In-pavement sensors (inductive, piezoelectric, and strain)	⊕ High detection accuracy with axle counting ⊖ Invasive and costly installation due to in-pavement integration	[8, 9]
Directional high-end radar sensors	⊕ Multilane multiobject traffic participant detection ⊕ Directional and velocity information ⊖ Cost and complexity of solutions	[10, 12, 13]
Top-down ultrasonic and radar sensors	⊕ Low cost and simple mode of operation ⊖ Only single-lane operation ⊖ Mounting above lane required, for example, on sensor gantry	[10, 14–18]
Proposed sidefire ultrasonic sensing	⊕ Multilane operation possible ⊕ Sidefire mounting allows seamless integration into environment ⊖ No directional information available in sidefire configuration	
User-based distributed traffic monitoring techniques		
Crowdsourcing systems (smartphones and navigation devices)	⊕ High number of devices ⊖ Privacy concerns ⊖ Susceptibility to attacks and data manipulation	[21–23]
V2X-based cooperative systems	⊕ Combination with in-vehicle information and decisions ⊖ Low support and prevalence	[24]

Camera-Based Systems. Frequently, camera-based systems are used for traffic monitoring. Here, we focus on systems with an automated analysis that is based on Computer Vision algorithms. With camera-based systems, a high detection accuracy plus a vehicle classification can be achieved. During nighttime, further solutions are required, for example, artificial illumination of the road and infrared or thermal imaging. Only in bad visibility conditions as fog, snow, or smog, the performance is limited [6]. The main concern with these systems, however, is the issue of privacy arising with video cameras in the public domain. On the one hand, the personal privacy of the citizens is affected; on the other hand, governmental regulations might even forbid the use of cameras in public areas, especially in this high-coverage case. Also, even with only a sparse coverage of video cameras in the public space, acceptance levels amongst citizens are low [7].

In-Pavement Sensors. Sensors in the urban area embedded into the pavement asphalt are often used in city intersection areas for traffic light control. The main in-pavement sensor technologies are inductive loops, piezoelectric sensors, and strain gauges [8]. They can also enable high-accuracy traffic measurements together with a classification by counting the axles of vehicles [9]. As a downside, the installation of pavement-integrated sensors is highly invasive and costly and requires per-lane road work with a possible temporary disturbance of city traffic.

Infrared Sensors. Passive and active infrared sensors are nonintrusive sensor types which can be mounted on the side or directly above the street. They offer detection with additional speed and vehicle class information in some realizations. With active sensors, multilane operation is possible. However, infrared sensors are susceptible to bad weather conditions like rain, snow, or fog. Furthermore, active infrared sensors require regular cleaning which might lead to lane closure during maintenance [10, 11].

Radar Systems. From the basic object detection principle, radar sensors represent the technology most comparable to ultrasonic sensors in the field of traffic sensing. High pulse repetition rates and ranges together with the possibility of obtaining directional information together with velocity data allow the monitoring of multiple traffic lanes and objects independent of the weather conditions [10, 12, 13]. As a major downside, the significant cost of the radar system, including front-end and signal processing, can compromise

a city-wide system deployment for a sufficient coverage with these sensors.

Top-Down Ultrasonic and Radar. In the field of radar and ultrasonic sensors, several top-down sensor solutions are available, which require the placement above a specific street lane for single-lane car profile measurements [10, 14–18]. These techniques allow obtaining the complete height profile of vehicles for detection and classification but require a gantry or bridge over the street for mounting the sensors, which is infeasible in many situations. Moreover, only a single lane can be monitored with the existing solutions [10]. For the ultrasonic sensors, the processing is strongly simplified, as with the available systems, only the distance to the first relevant reflection and not the complete impulse response is evaluated. The significantly lower propagation speed and resulting decay time of the impulse responses together with the requirements for a sufficient pulse repetition rate (typically ≥10 Hz) of ultrasonic sensors can lead to self-interference and also to issues with fast-moving traffic participants [18]. For ultrasonic sensors, previous studies have shown that the impact of weather effects such as wind, snow, and rain is only marginal [19, 20].

Crowdsourcing Systems. For the second category of end-user based distributed information systems for traffic monitoring, one of the most present techniques is the crowdsourcing of traffic monitoring to end-user smartphones and navigation devices in order to obtain real-time and high-density traffic flow information [21, 22]. This approach is used in several systems and solutions, for example, by "Google Maps" with the product "Google Traffic" or by "TomTom HD Traffic." As a central advantage, no investments in additional sensors are required and the very high prevalence of these applications on end-user devices can be exploited by actively tracking the devices. However, the live tracking of the users is highly critical in terms of privacy aspects. Furthermore, these systems are susceptible to manipulative techniques and fake information due to trust issues with the user base, which has been demonstrated before [23].

V2X Communications. With the emerging field of V2X (vehicle-to-everything) communications, which is currently focused on on V2V (vehicle-to-vehicle) and V2I (vehicle-to-infrastructure) techniques, new possibilities for traffic information acquisition and exchange arise. Together with in-vehicle information for a direct delivery of inferred routing decisions to the driver and in a fusion with available travel plans on the car navigation device, an optimized information exchange and routing is enabled. Nevertheless, these types of systems are still to be realized in mass market together with suitable application scenarios and it will take time until they are sufficiently established amongst everyday vehicles. Moreover, reservations with regard to privacy and trust are also an important aspect for these systems. In literature, it is shown that, with a sufficient prevalence of cooperative V2V techniques in cars together with a high relaying communication range, dense traffic information can be obtained [24]. Therefore, vehicular communication techniques are a promising approach for the future. Certainly, as long as the density and proportion of vehicles supporting V2X communications is too low, a diverse traffic monitoring approach, which in parallel comprises infrastructural sensing techniques, has to be pursued for several decades.

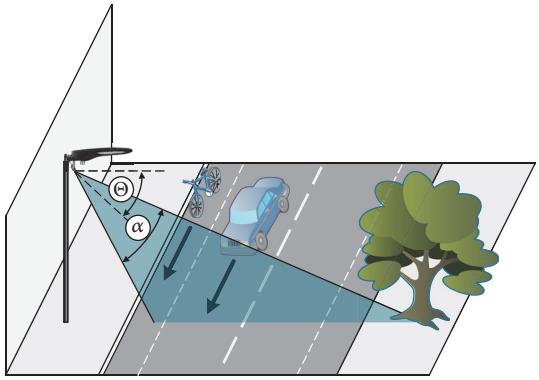
2.2. Specific Advances in Ultrasonic Traffic Detection. In the previous introductory Section 2.1, the basic top-down ultrasonic sensing technology has already been introduced. However, the practical use is limited, as additional mounting efforts are required, if no infrastructure like bridges above the street or traffic light poles at intersections are available. Moreover, only single-lane detection is possible in this regard. Therefore, these solutions are limited in their practical applicability.

There also exist initial concepts that use ultrasonic sensors mounted on a lateral street position with a so-called sidefire orientation, facing the opposite street side either horizontally or in a downtilted position [19, 25, 26]. In the case of horizontal tilt, even a first simple multilane detection has been realized. However, all these concepts perform only an analysis of the first-reflection distance instead of the whole impulse response, which highly limits the practical use in a variety of scenarios. The urban scenario is a highly reflective time-varying environment, where first reflections of other objects such as moving trees, parking cars, and the scattering of the road surface are always present in the distance range with the relevant traffic participants for detection, the distance region-of-interest for measured signal reflections of objects. In addition, the horizontally facing type of sidefire sensors could be obstructed by other objects and traffic participants in urban scenarios due to their low mounting height.

3. System Concept and Architecture

Sensor systems have specific requirements regarding the setup, scenario, and working environment they are being operated in. In this chapter, the system setup and application concept, together with the generic requirements for our proposed ultrasonic traffic sensing system, are given. The scenario stays the same as initially proposed in Section 1: a Smart City street lamp system platform. Nevertheless, a wide variety of installation and application possibilities are given also without any ties to this integration concept.

3.1. Sensor System Setup. An exemplary setup of the smart street lamp platform was already shown with Figure 1. It features a single mounted ultrasonic sensor attached to the lamp head or the pole and an embedded system integrated into a customized LED-based lamp head. The digital core system contains modules for realizing processing and communication capabilities, enabling distributed processing in a local-level group of lamps in a street by building a dynamic mesh network and also allowing communication on the global and cloud level by employing mobile network machine-to-machine type communication (M2M) functionalities. The central component for the signal processing capabilities consists of a Texas Instruments Sitara system-on-chip (SoC) including an ARM Cortex-A8 core. Typical examples of



Θ: Downtilt angle
α: Beam width

Figure 2: Downtilted sidefire sensor configuration in typical urban environment with two-lane street (with downtilt angle Θ and beam width α).

possible applications with this system are location-based content delivery, attachment of modules for environmental or traffic sensing applications, and energy efficiency optimizations of street lighting. Due to the combination of infrastructure with an extensible platform, the systems can be widely distributed amongst the city, as almost all central areas are covered by street lamps. Furthermore, many of the constraints on processing and communications, which are typically imposed for Smart City traffic monitoring systems in IoT context, can be relaxed, for example, data and refresh rate restrictions due to battery power [5].

The typical seamless integration of the ultrasonic sensing concept into an urban street area is shown in Figure 2. In the so-called downtilted sidefire setup, the sensor is mounted in a height between 3 and 8 m, which is above the typical height of vehicles in urban areas. The sensor head is then oriented diagonally downwards the street, with a downtilt angle Θ between 30° and 80°. The ultrasonic sensor transducer together with the casing has a cone-shaped beam characteristic, with an opening angle α typically between 20° and 80° depending on the scenario and lane coverage requirements. In the exemplary scenario shown in Figure 2, the multitude of reflective elements in the urban environment already becomes apparent. For example, the first reflections and scattering of the street surface can arrive earlier than reflections from relevant objects on distant lanes to the sensor. Additionally, reflective objects can exhibit short- or long-term time variation, like parked cars or trees moving in the wind. In these cases, simple distance-measurement based sensor techniques would already fail.

3.2. Ultrasonic Sensor Platform Module. In order to provide the capabilities for ultrasonic sensing, the core system is extended with a custom sensor platform module, which is shown in Figures 3 and 4. It provides four ultrasonic sensor channels for real-time arbitrary waveform generation and recording in half-duplex mode. The reception quality of the ultrasonic impulse response is with an input stage SNR of above 105 dB in the ultrasonic band. The analog driving circuit for the transducer is integrated into the ultrasonic transducer head itself and is adapted to the capabilities and characteristics of the transducer.

Control of the attached components, signal generation, and preprocessing capabilities is realized with an integrated Xilinx Artix-7 series FPGA. It provides communication capabilities to the core processing system over an Ethernet link, which is also used for the power supply of the sensor system extension. Together with the core system, GPS including the high-precision PPS signal is used for a timing synchronization and exact coordination of multiple sensors in the vicinity of a street. Interference can therefore be coordinated by a global optimization of the sensor timing patterns if the sensor heads are transmitting in the same frequency band.

3.3. General Signal Structure. For simplicity, the signal structure in the single-sensor case is discussed. The transmit signal $s_T(t)$ consists of repeated sequences of the base pulse $h_i(t)$ with an overall duration T_{rep}, yielding a pulse repetition rate of $f_{\text{rep}} = T_{\text{rep}}^{-1}$. This is also shown in Figure 5. T_{rep} is typically chosen between 50 and 150 ms to ensure low self-interference levels between the different impulse response blocks. The repetition time should in general be larger than the RT60 reflections decay time of the acoustic channel [27, p. 98]. The RT60 time is a common measure in indoor acoustics but can also be applied in outdoor scenarios, where it typically yields much shorter values. The general pulse sequence is then defined as follows:

$$s_T(t) = \sum_{n=0}^{\infty} h_n\left(t - n \cdot T_{\text{rep}}\right). \tag{1}$$

The base pulses $h_i(t)$ itself consist of two specific duplexer phases. At the beginning, the transmit duplex mode is used and a transmit pulse part of length T_{wnd} is emitted. Then, the duplex mode is switched to receive mode and records the channel impulse response for the rest of the period $T_{\text{rep}} - T_{\text{wnd}}$ without further transmission. For the investigations in this article, the transmit pulse is chosen to be a windowed signal with a carrier frequency of $f_c = 40\,\text{kHz}$ and an envelope window $h_{\text{wnd},i}(t)$ with length $T_{\text{wnd}} = 2\,\text{ms}$ as a tradeoff between time and frequency resolution for the narrowband signal. The i-th base pulse is then given by the following:

$$h_i(t) = \begin{cases} h_{\text{wnd},i}(t) \cdot \text{Re}\left\{e^{j2\pi f_C t}\right\} & \text{if } 0 \le t \le T_{\text{wnd}}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

With the use of identical repeated transmit pulses in our case, $h_i(t) = h(t)\ \forall i \in \mathbb{N}$. A Blackman window [28] function is chosen for $h_{\text{wnd},i}(t)$ because it does not exhibit discontinuities at the beginning and end of the time domain window, while exhibiting a high stop-band attenuation [29].

4. Processing Concept and Algorithms

The ultrasonic traffic sensing technique presented in this paper is based on a combination of statistical analysis

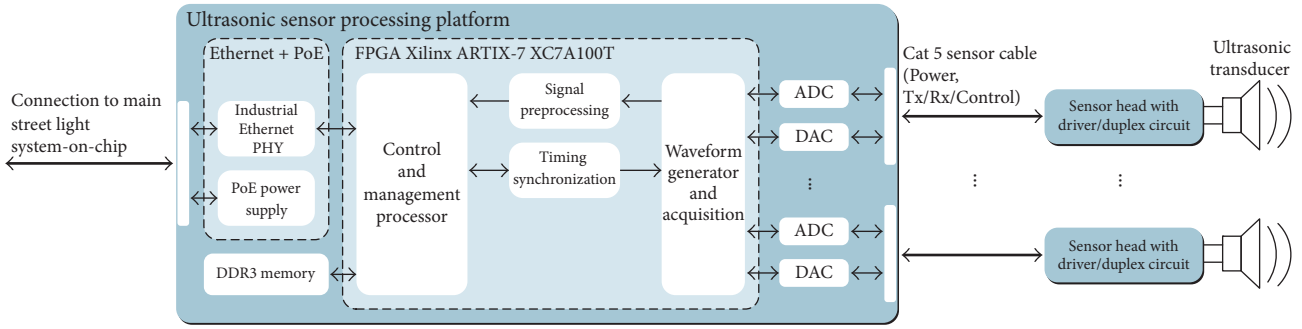


Figure 3: General structure of sensor platform including the FPGA and attached ultrasonic sensors with driving circuits.

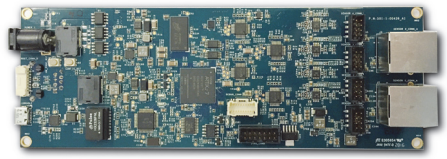


Figure 4: Custom sensor platform which is integrated as an extension module into the intelligent street light.

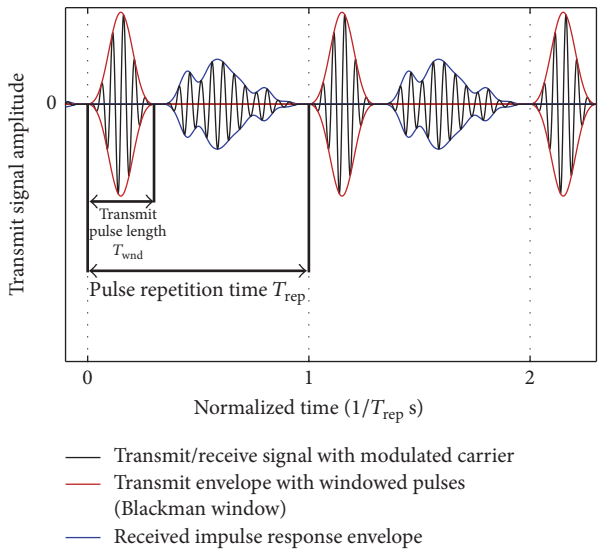


Figure 5: Generalized signal structure with transmit/receive duplexing mode, corresponding envelopes, and modulated signals.

and clustering techniques for object candidate detection. Together with a special signal chain structure, objects representing traffic participants can be detected, together with an extraction of further characteristics and object properties. In this section, the model for the received signal is given first. Then, the characteristics of the reflective environment are modeled and empirically analyzed. Based on these findings, statistical hypothesis testing is performed during operation and the resulting statistical information is combined with a modified clustering algorithm for object boundary detection. Then, the system concept for the extraction of further characteristics is given and the reduced-complexity embedded implementation for real-time operation is discussed.

4.1. Received Signal Model and Preprocessing. With the general signal structure described in Section 3.3, repeated measurements of the acoustic channel are performed with the ultrasonic sensor system. Based on the impulse responses as the one-dimensional received signals, which consist of a large number of reflections in the acoustic environment, object detection and further analyses can be performed together with the time-of-flight distance information.

The received signal $s(t)$ from the sensor head is recorded continuously and then sliced into blocks of length T_{rep}, yielding a series of impulse responses $s_i(t_D)$, with i being the ith measurement interval of length T_{rep}. The parameter t_D is the time position of the specific impulse response, which is mapped to an equivalent time-of-flight distance $d = t_D \cdot c$ given the speed of sound $c = 343\,\text{m/s}$ in the air medium at normal conditions. Due to the round-trip, the real distance to the reflective object is half the equivalent time-of-flight distance. By acquiring information on the current air temperature via sensor measurements or weather information, c can be compensated for the current conditions. During the initial period of $s_i(t)$ with $0 \leq t_D \leq T_{\text{wnd}}$, which is the transmission duplex mode, no useful signal is acquired, while the region $T_{\text{wnd}} \leq t_D \leq T_{\text{rep}}$ represents the main signal of interest for further analysis. The single impulse response slices are then given by the following:

$$s_i\left(t_D\right) = s\left(i \cdot T_{\text{rep}} + t_D\right). \tag{3}$$

The equivalent time-discrete representation is

$$x_i\left(n_D\right) = s\left(i \cdot N_{\text{rep}} T + n_D \cdot T\right). \tag{4}$$

with the sampling time T and the discrete-time repetition interval length $N_{\text{rep}} = T_{\text{rep}}/T$.

As a first step of preprocessing of the sliced receive signal on the FPGA, each impulse response is convolved with a bandpass filter $h_{\text{BP}}(t)$ centered at the transmit carrier frequency f_C with a bandwidth of 6 kHz for allowing the analysis of Doppler-shifted signals with typical urban vehicle velocities. Then, a time-discrete Hilbert transform $\mathscr{H}$ [30, 31]

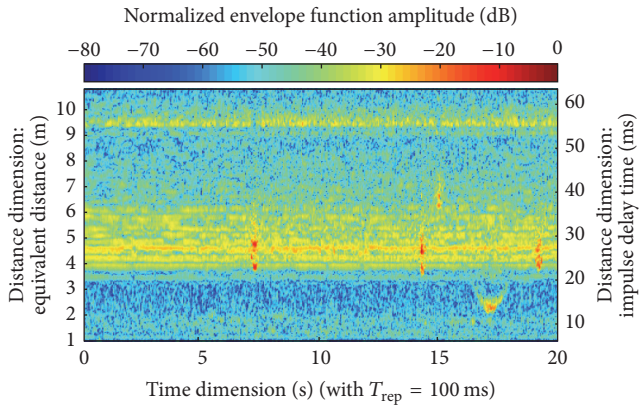


FIGURE 6: Two-dimensional envelope function mapping of received signal $e_{R,i}(n_D)$ (amplitude normalized and in logarithmic scale).

is performed in order to acquire the complex-valued analytic signal:

$$x_i(n_D) = (s_i * h_{BP})(n_D) + j \cdot \mathscr{H}\{(s_i * h_{BP})(n_D)\} \quad (5)$$

together with the instantaneous amplitude or real-valued envelope signal

$$e_i''(n_D) = |x_i(n_D)|. \quad (6)$$

This envelope signal $e_i''(n_D)$ is further processed to facilitate object detection in the following. Only in case of Doppler-based motion and velocity estimation, time-frequency information such as the instantaneous phase and frequency has to be incorporated. Both its two discrete dimensions, i being the index of the impulse response (from now on called time dimension, with an equivalent sampling rate of f_{rep}) and n_D being the distance-equivalent impulse response time (from now on called distance dimension, with an equivalent sampling rate $f_S = 1/T$), are derived from a single original time dimension in the received signal by slicing. Therefore, the distance dimension is fixed and limited to $0 \le n_D \le N_{rep} - 1$, while the causal time dimension in real-world application is extending with f_{rep}. With this two-dimensional mapping, object detection algorithms become applicable which consider a vicinity in both dimensions. This means that effects of an object creating a peak at several neighboring distance and time points are now properly represented as neighbors in the 2D space, in contrast to the original one-dimensional recorded sensor signal.

An example for the envelope signal $e_{R,i}''(n_D)$ is given in Figure 6, with a normalized logarithmic color mapping of the amplitude for improved visualization. The equivalent distance and impulse response time mappings are shown on the ordinate for reference purposes. In this plot, the scattering and reflections from the street surface can already be seen in a distance range of 4 to 6 m in a comb-like pattern. Also, around the time positions at 7 s, 14 s, 15 s, 17 s, and 19 s, strong reflections from objects passing the sensor are visible.

4.2. Signal Statistics and Standardization. Given the high complexity of the reflective patterns in the sensor data, the statistics of the signal need to be described properly in order to perform an outlier or anomaly detection, which would indicate the presence of objects in the sensor field. The goal is to incorporate all noise components (measurement noise and other acoustic emissions in the ultrasonic band) and also the typical static and time-varying reflections (e.g., moving leaves of a tree caused by wind and reflection patterns of the street) for every specific distance point of the signal into the statistics. This allows the classification of segments of newly acquired impulse responses in terms of the data following the a priori distribution, called base distribution, which was estimated in the past, or whether an outlier is present.

In contrast to typical binary decision problems, here, only this base distribution without any presence of an object can be estimated, representing the null hypothesis. The alternative hypothesis of object presence is however different for every object. This problem can be solved using parametric or nonparametric hypothesis testing techniques for properly representing the underlying base distributions of the signal points and testing the outlier probabilities. However, for the real-time implementation on the embedded system of the sensor setup, we present a simpler and less computationally complex approach that is based on the distance-wise standardization of the signal based on the calculated statistics in a predefined time window in the past. These standardization techniques are often used in the field of data science as a preprocessing step for feature scaling. The distance-wise standardized envelope signal $e_i'(n_D)$ is calculated based on the information from the statistics memory with a window length of $N_{T,w}$ time dimension steps in the past (equivalent to a time range of $N_{T,w} \cdot T_{rep}$ in continuous time), yielding $e_i'(n_D)$:

$$e_i'(n_D) = \frac{e_i''(n_D) - \mu_{n_D}(i)}{\sigma_{n_D}(i)} \quad (7)$$

$$= \frac{e_i''(n_D) - \overbrace{N_{T,w}^{-1} \sum_{k=i-N_{T,w}}^{i} e_k''(n_D)}^{\text{Windowed mean } \mu_{n_D}(i) \text{ for distance } n_D}}{\underbrace{\sqrt{(N_{T,w}-1)^{-1} \sum_{m=i-N_{T,w}}^{i} \left[e_m''(n_D) - N_{T,w}^{-1} \sum_{k=i-N_{T,w}}^{i} e_k''(n_D)\right]^2}}_{\text{Windowed standard deviation } \sigma_{n_D}(i) \text{ for distance } n_D}}, \quad (8)$$

$$0 \le n_D \le N_{rep} - 1, \; i \ge N_{T,w}.$$

Under the simplified assumption that the underlying process of $e_i''(n_D)$ (with no objects present) for a given distance n_D is an i.i.d. Gaussian process $X_{n_D} \sim \mathcal{N}(\mu_{n_D}, \sigma_{n_D}^2)$, the specific distance ranges of $e_i'(n_D)$ will follow the standard normal distribution $Y_{n_D} \sim \mathcal{N}(0,1)$. The assumption is especially found to be valid for direct reflections in the closer range, shown in studies from other acoustic environments like underwater [32]. As a general approximation, it will later be used for approximatively calculating the initial detection thresholds for a given false alarm rate.

For the object detection, only the right tail of the distribution is of interest for outlier detection, as shadowing effects are not considered. Therefore, the complete standardized distribution data $e_i'(n_D)$ is clipped for values below zero,

resulting in the final envelope $e_i(n_D)$ with statistical standardization:

$$e_i(n_D) = \max\left(e_i'(n_D), 0\right). \tag{9}$$

This yields a variable with a standard normal distribution left-censored at 0. The resulting probability distribution function (PDF) $f(x)$ can be written as follows:

$$f(x) = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} + \frac{1}{2}\delta(x) & x \geq 0, \\ 0 & \text{otherwise;} \end{cases} \tag{10}$$

and the cumulative distribution function (CDF) $F(x)$ is

$$F(x) = \begin{cases} \frac{1}{2}\operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) + \frac{1}{2} & x \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $\operatorname{erf}(x)$ is the Gauss error function. The first terms of $f(x)$ and $F(x)$ are similar to a scaled half-normal distribution [33], with the addition of the clipped values being statistically censored and not truncated.

To summarize, $e_i(n_D)$ is now used for all further object detection and processing together with the clustering techniques in the following. Simply put, it yields the positive outlier level of a newly acquired data point by the distance to the calculated mean in a measure of the multiple of standard deviations. This is based on the statistical history in a specific time interval and calculated separately for each discrete distance point. By having a limited time window for the statistics, the sensor system can adapt to changing environments without needs for recalibration. Optionally, for achieving a better robustness and generalization, these distance-wise statistics can be blurred with neighboring distance points.

4.3. Object Boundary Detection with Density-Based Statistical Clustering (DBStaC). As a next step, the standardized data given in the previous section is used for object detection. The two-dimensional data $e_i(n_D)$ is passed through a modified density-based clustering algorithm based on the original DBSCAN (Density-Based Spatial Clustering of Applications with Noise [34]), which is able to perform clustering on noisy data by aggregating core points of data peaks. In our case, these data peaks can be the result of statistical outliers due to physical objects passing by the sensor, in comparison with the base distribution of the signal when no objects are present. Our proposed modified algorithm in combination with the input preprocessing steps from (7) and (9), called *density-based statistical clustering* (DBStaC), can be used on both statistical hypothesis test results, and in our case of a simplified algorithm, data standardized based on the a priori statistics.

4.3.1. Choice of Clustering Algorithm. The choice of a suitable clustering algorithm from the field of unsupervised learning techniques for our application was subject to several requirements. While many clustering algorithms can aggregate nearby points in a space with arbitrary dimensionality and sparsely distributed data points, it is required here to incorporate the weight of data points, coming from our the nonsparse standardized and clipped data field $e_i(n_D)$ described before in Section 4.2.

Furthermore, the algorithm should be able to detect important data peaks based on a local density, while, for the whole cluster, no penalty for an infinite extension in the time dimension should be given. This is necessary because the dwell time of an object in the sensor range is arbitrary, especially due to different movement speeds of traffic participants or even in a traffic jam situation.

As a third requirement, an anisotropy of the cluster properties and shape is desired, as we have two dimensions that are time (index of different impulse responses) and distance (specific point in the impulse response itself), which have highly different characteristics. A typical scenario where anisotropy would not be required is 2D image processing, where both image dimensions have similar properties.

4.3.2. Original Definition of DBSCAN. First, we want to provide a basic understanding of the original DBSCAN algorithm in general and the specific terms coming with the algorithmic definition. Then, we can introduce the modifications for our specific application. Readers interested in the complete formal description are referred to [34, 35], on which the following original definitions are based.

Core Point. The general principle of DBSCAN is the analysis of a set D of points in a k-dimensional space. Then, for any given point $\mathbf{p} \in D$, the ε-neighborhood of this point is evaluated, meaning that all points $\mathbf{q} \in D$ with a distance $\operatorname{dist}(\mathbf{p}, \mathbf{q}) \leq \varepsilon$ belong to the neighborhood of $\mathbf{p}$, denoted $\overline{N}$. The distance metric $\operatorname{dist}(\mathbf{p}, \mathbf{q})$ is typically chosen to be the L_1 or L_2 norm, and ε is preset. This results in the score $\overline{n}_\varepsilon$ of a point $\mathbf{p}$ being the number of other points $\mathbf{q}$ falling into the ε-neighborhood. Then, the point $\mathbf{p}$ is called core point if $\overline{n}_\varepsilon \geq \text{MinPts}$, with MinPts being a preset threshold for core point detection. All points in the ε-neighborhood of a core point are part of a cluster set C, and they are called border points if they are not core points themselves.

Direct Density-Reachability. A point $\mathbf{p}$ is directly density-reachable from a point $\mathbf{q}$ if $\mathbf{q}$ is a core point and $\mathbf{p}$ is in the ε-neighborhood of $\mathbf{q}$. This property is symmetric if $\mathbf{p}$ and $\mathbf{q}$ are a pair of core points [34].

Density-Reachability. A point $\mathbf{p}$ is density-reachable from a point $\mathbf{q}$ if a chain of n points $\mathbf{p}_1, \ldots, \mathbf{p}_n$ with $\mathbf{p}_1 = \mathbf{q}$, $\mathbf{p}_n = \mathbf{p}$ exists where every point $\mathbf{p}_{i+1}$ is directly density-reachable from $\mathbf{p}_i$, $\forall i \in [1 \cdots n-1]$ [34].

Density-Connectivity. A point $\mathbf{p}$ is density-connected to another point $\mathbf{q}$ if there is a point $\mathbf{o}$ such that both $\mathbf{p}$ and $\mathbf{q}$ are density-reachable from $\mathbf{o}$ [34].

Cluster Definition. Given the set of all points D, a cluster C is a nonempty subset of D which satisfies the following conditions [34]:

(1) $\forall \mathbf{p}, \mathbf{q}$: if $\mathbf{p} \in C$ and $\mathbf{q}$ is density-reachable from $\mathbf{p}$, then $\mathbf{q} \in C$ (maximality).

(2) $\forall \mathbf{p}, \mathbf{q} \in C$: $\mathbf{p}$ is density-connected to $\mathbf{q}$ (connectivity).

Any cluster C is then uniquely determined by its core points.

4.3.3. Modified Core Point Definition for DBStaC. In our special case, the algorithm is modified to suit the needs for clustering our two-dimensional space; therefore $k = 2$ in order to represent the time and distance dimension of our acquired data. Furthermore, our data points are weighted by their standardized value. Therefore, for a point $\mathbf{p} = (i, n_D)$, the weight $e_i(n_D)$ is assigned, which is not an option in the classic DBSCAN algorithm. Instead of using the dist metric for determining the ε-neighborhood, anisotropy is introduced by choosing the ε-neighborhood as a rectangular area with dimension $(2\varepsilon_T + 1) \times (2\varepsilon_D + 1)$, symmetrically placed around the position (i, n_D) of the core point, with ε_T and ε_D given. Then, all points $\mathbf{q}(i', n_D')$ with $|i' - i| \leq \varepsilon_T \wedge |n_D' - n_D| \leq \varepsilon_D$ belong to the ε-neighborhood N of $\mathbf{p} = (i, n_D)$ and $\mathbf{p}$ is a core point if

$$\sum_{t=i-\varepsilon_T}^{i+\varepsilon_T} \sum_{d=n_D-\varepsilon_D}^{n_D+\varepsilon_D} e_t(d) \geq \text{MinSum}. \tag{12}$$

With this definition, all data points of our two-dimensional data set are now taken into account, not just by their count, but by the sum of their assigned weights in the specific ε-neighborhood. The implementation of the final clustering is omitted at this point and widely described in literature [36]. With the original algorithm implemented, adaptions only have to be made in the core point definition and neighborhood metrics.

4.3.4. False Alarm Rate. In the clustering process, every element of $e_i(n_D)$ is tested for the core point property, requiring both compensation for multiple comparisons and an approximation of the false alarm rate for a single point itself. For the latter, we will provide measures for determining the false alarm rate under the approximative distribution assumptions from Section 4.2. With the threshold decision from (12) for a core point, given MinSum and the ε-neighborhood with

$$A = (2\varepsilon_T + 1)(2\varepsilon_D + 1) \tag{13}$$

points in the rectangular region, we want to calculate the false alarm probability:

$$P_f = P\left(S(i, n_D) \geq \text{MinSum} \mid \overline{O}\right), \tag{14}$$

where $\overline{O}$ is the event that no relevant object reflection yielding a core point is present, meaning that all summands follow the base distribution without outliers. $S(i, n_D)$ is the sum of weights in the ε-neighborhood:

$$S(i, n_D) = \sum_{t=i-\varepsilon_T}^{i+\varepsilon_T} \sum_{d=n_D-\varepsilon_D}^{n_D+\varepsilon_D} e_t(d). \tag{15}$$

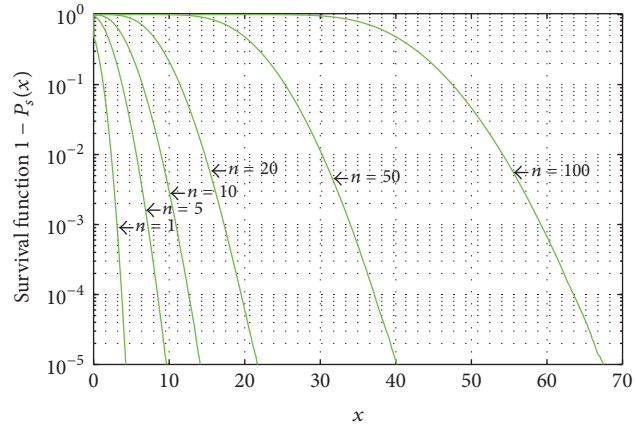


FIGURE 7: Survival function $1 - P_s(x)$ for distribution $p_s(x)$ with n-fold convolution of $f(x)$.

In Section 4.2, we showed that under the assumption that the original envelope signal $e_i''(n_D)$ is coming from a process with normal distribution, $e_i'(n_D)$ is standard-normally distributed after standardization, and $e_i(n_D)$ follows a left-censored standard normal distribution. If the data points $e_i(n_D)$ $\forall i, n_D$ and the resulting $S(i, n_D)$ are also hypothesized to be statistically independent and exhibit the identical PDF $f(x)$ from (10), we can write for the resulting PDF $p_S(x)$ of the summation of $n = A$ points in (15):

$$p_S(x) = \underbrace{(f * \cdots * f)}_{n \text{ times}}(x) = f^{*n}(x). \tag{16}$$

Please note that the statistical independence of all $S(i, n_D)$ is not given in the first place because the same data points in the two-dimensional field are affected by the sliding window in several i and n_D positions. Then, adjacent $S(i, n_D)$ are correlated and a single peak in $e_i(n_D)$ could yield multiple peaks in $S(i, n_D)$ resulting in core points. However, as we seek to calculate the false alarm rate in general for this specific algorithm, adjacent core points would be clustered into a single cluster, raising only false alarm for a single object if $P_f \ll 1$.

Finally, neither for the left-censored standard normal distribution $f(x)$ nor for the related half-normal distribution in general, the result of k-fold convolution is known or given in literature in closed form for $k > 2$. Therefore, the final calculation is based on Monte Carlo simulations of these random processes. For the false alarm probability analysis, we are interested in the survival function (SF) $1 - P_s(x)$, where $P_s(x)$ is the CDF of $p_s(x)$. With $x = \text{MinSum}$, the survival function yields the false alarm probability per data point. The results from the Monte Carlo simulations for the survival function are shown in Figure 7 and were also verified with an additional false alarm Monte Carlo simulation.

4.3.5. Choice of Clustering Neighborhood. The neighborhood choice in the time direction, ε_T, affects the detection robustness and separability of vehicles with length l_{Veh} following each other in a gap distance l_{Gap}, with speed v and passing the sensor lobe which approximately covers a section of length

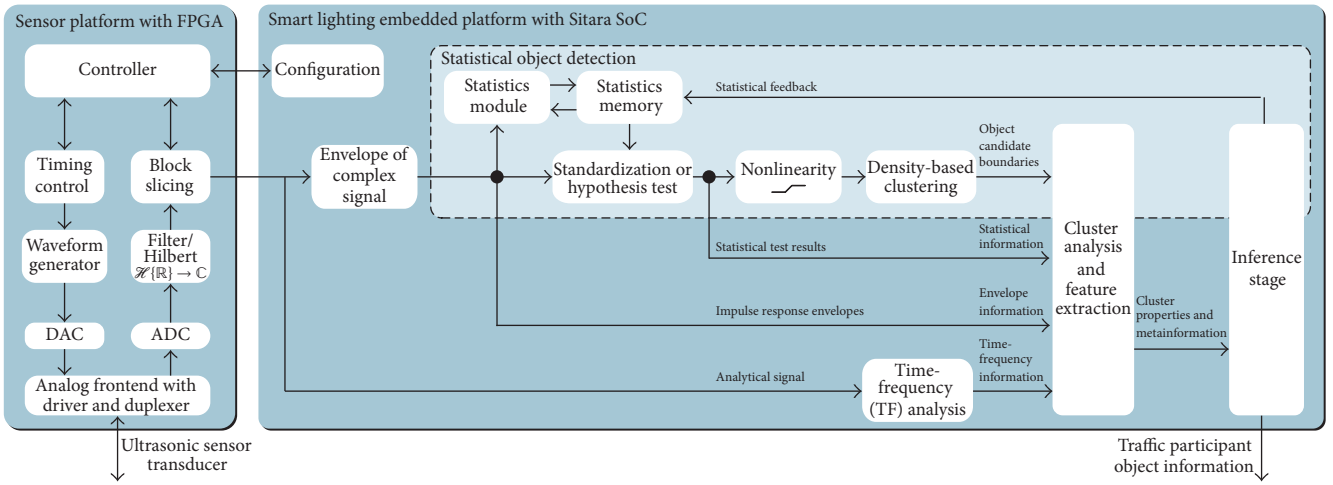


Figure 8: Signal chain with acquisition, analysis, and inference stages on both platforms.

l_{Lobe} on the specific lane. Given the pulse repetition rate T_{rep}, no vehicle is covered by the sensors during the gap for n_{Gap} pulse times:

$$n_{\text{Gap}} = \frac{\left(l_{\text{Gap}} - l_{\text{Lobe}}\right)}{\left(v \cdot T_{\text{rep}}\right)}. \tag{17}$$

Also, considering the detection of a single vehicle, the vehicle is sampled in the lobe for n_{Veh} pulses:

$$n_{\text{Veh}} = \frac{\left(l_{\text{Veh}} + l_{\text{Lobe}}\right)}{\left(v \cdot T_{\text{rep}}\right)}. \tag{18}$$

Given ε_T, the full neighborhood extension in time direction is $2\varepsilon_T + 1$ pulses. Therefore, $n_{\text{Gap}} \geq 2\varepsilon_T + 1$ is recommended for a good separation of the vehicle clusters in time direction in order to have no overlapping core points. Typical values of the time neighborhood are $0 \leq \varepsilon_T \leq 2$. With an urban area example of $\varepsilon_T = 1$, $l_{\text{Veh}} = 4.5$ m, $v = 40$ km/h, and $l_{\text{Lobe}} = 1.5$ m, the recommended gap length becomes $l_{\text{Gap}} \geq 4.83$ m and a vehicle is covered by $n_{\text{Veh}} = 5.4$ pulse times. The second neighborhood parameter ε_D in distance direction is mostly relevant when it comes to multilane separation of multiple vehicles. In general, our approach in this paper is to learn the typical vehicle speed and distance profile without a direct calculation, but by learning the scenario for a short training period (e.g., one hour) and then setting the neighborhood parameters in a parameter optimization, which is described in the following section.

4.4. Signal Processing Concept. In Figure 8, the previously described statistical object detection is integrated into the complete object detection concept with feature extraction and inference. It allows detecting candidates for detected objects representing traffic participants, which are then used to gather further information from several signal processing paths. The preprocessing and main processing stages are also divided in hardware into the sensor platform FPGA and the Sitara SoC, respectively.

4.4.1. Preprocessing. In contrast to the sensor platform structure described previously in Section 3, here, the signal flow and processing are in focus. As can be seen in the left part of Figure 8, the sensor platform is configured by the main system on the Sitara SoC and performs the whole signal transmission and reception. The preprocessing steps of filtering, Hilbert transformation, and possible sample rate adjustments are performed using a dedicated architecture on the FPGA, before slicing the blocks from the different sensors into single impulse responses, which are then transmitted to the main processing system with the Sitara SoC.

4.4.2. Object Detection and Extraction. On the right side of Figure 8, the complete object detection and feature extraction concept is shown. It is based on four main information paths going into the cluster analysis and feature extraction block at the end, whose output is then used for inference of detected traffic participants. By performing a fusion of these four paths, all necessary information for the analysis can be provided. The different components of the processing system are described in the following.

Statistical Object Detection with Resulting Object Candidate Boundaries. The basic principle of object clustering on the standardized data for boundary detection has already been described before in Sections 4.2 and 4.3. The only addition is an optional nonlinearity after standardization or the optional hypothesis testing (which is not considered here). The nonlinearity can be modified to perform further transformations on the standardized data. With the DBStaC algorithm, the envelope signal at the beginning of the statistical object detection block is analyzed by the statistics module together with its statistics memory. Only a predetermined amount of past data can be stored in this fixed-size memory for calculating the base statistics. The boundaries of resulting clusters of the algorithm are taken as boundaries for further analyses in the time-distance-field of the different data paths. Inside each cluster's area which was determined by the object boundary information path, the cluster analysis stage

performs an analysis of the statistical, envelope, and time-frequency-information path.

Statistical Feedback. The statistical feedback path coming from the inference stage is important to keep the base distribution in the statistics memory accurate to be mostly restricted to data without objects present. This is an optional feature and especially helpful with high fluctuations of traffic density in front of the sensor, as the detected objects are removed from the statistics memory and the outlier analysis of the DBStaC-based object detection is more accurate and robust against drift.

Statistical Information and Envelope Information. The standardized data and the original envelope signal are directly fed into the cluster analysis stage. For feature extraction and accurate information on the object reflection characteristics, properties like the accurate position of the object's first reflection and the mathematical two-dimensional center of mass of the signal or the geometry are analyzed. Especially with a priori geometric information on the system setup in a specific scenario, effects caused by the beam width of the transducer can be exploited. Even if a car moves perfectly sideways through the cone-shaped beam of the sensor, it forms a typical arch-shaped pattern, which could already be seen in the initial example signal, Figure 6. This arc-shaped reflection pattern is also known from marine sonar systems as "fish arc" and caused by the fact that at the point in time when a vehicle first enters the beam, the reflection path is diagonal and is not fully straight until the car is directly in front of the sensor. This can be used to support the information on vehicle size and type.

Time-Frequency Information. The time-frequency information is mostly irrelevant as long as the sensor is oriented perpendicular to the vehicle movement on the street. However, in case that the sensor is oriented partially sideways or a second sensor with such orientation is available, velocity information can be gathered by analyzing the instantaneous frequency of the signal for Doppler shift estimation.

Inference Stage. The inference stage performs the final decision whether the cluster detected in the DBStaC stage originated from a traffic participant in the sensor range and allows the estimation of further object parameters and a simple classification of the vehicle type using a supervised learning stage with a Support Vector Machine. As our analyses in this article are focused on the DBStaC object candidate detection algorithm itself, no further rejection of objects is performed by the inference stage in this case. Only objects detected outside of the specific lanes, for example, on the sidewalk, are rejected.

4.5. Implementation Details. The main parts of the processing chain shown on the right side of Figure 8 were implemented in C++, including optimized versions for embedded system operations, while parts of the postprocessing and learning techniques as part of the cluster analysis and inference stages were implemented in Python. The core processing module can be used for both algorithmic and performance analysis purposes on general Linux/BSD systems and the embedded Linux system running on the ARM Cortex A8 as part of the Sitara SoC. For algorithmic evaluation and parameter optimization in the next section, capabilities of the core module will also be used as part of an evaluation system. The numerical precision requirements for embedded system operation were relaxed from 64-bit floating point to 32-bit floating point, largely without compromising performance, in order to exploit the NEON floating point accelerator of the Sitara SoC. For the processing chain with the DBStaC algorithm, real-time operation is then possible for two attached ultrasonic sensors. This is the case for typical parameter sets, as the algorithmic runtime of DBSCAN is not fixed and is largely dependent on the choice of parameters (size of ε-neighborhood and decision thresholds) and traffic situation yielding different cluster sizes. Edge cases like extremely large clusters, for example, due to a traffic jam situation with very long dwell times of an object in the sensor range, have to be taken care of in the practical implementations.

For the sensor platform with the FPGA shown on the left side of Figure 8, dedicated accelerators for filtering and processing are used together with a MicroBlaze soft core for the coordination of waveform transmission and data acquisition. The configuration is controlled by the main processing system side with the Sitara SoC. A timing reference signal is acquired using GPS together with the high-precision PPS reference, which allows global synchronization of multiple sensors attached to multiple platforms, with an optimization of the transmission and interference patterns. Exchange of resulting information, for example, for sensor fusion and distributed processing for trajectory calculation of objects passing by multiple systems, is possible on both the level of a local wireless meshing between the systems and a global communication using cellular networks.

5. Evaluation Scenarios and Methodology

Evaluation of the complete system concept for traffic participant detection requires analyses both in real-world scenarios and in isolated conditions with synthetic scenarios, such as on automotive test tracks. A variety of European-wide test measurements have been performed with the system, described in Section 5.2, which will be analyzed and discussed in Section 6. For the reference data acquisition, a video camera is running in parallel which is afterwards used to label, annotate, and classify traffic participants on the time line. These labels can then be utilized to evaluate the performance in terms of several metrics for quantifying detection performance, parameter estimation, and classification. In general, it is desirable to have specific correctness information for every object instead of just taking the overall numbers of detected objects for the analysis, which is supported by an exact pairing of detected objects and reference data in the following. Then, parameter sets for the system are optimized with regard to different performance metrics using a central (hyper)parameter optimizer as part of an evaluation framework, which is described in the following. This process can also be deployed to a high-performance cluster (HPC).

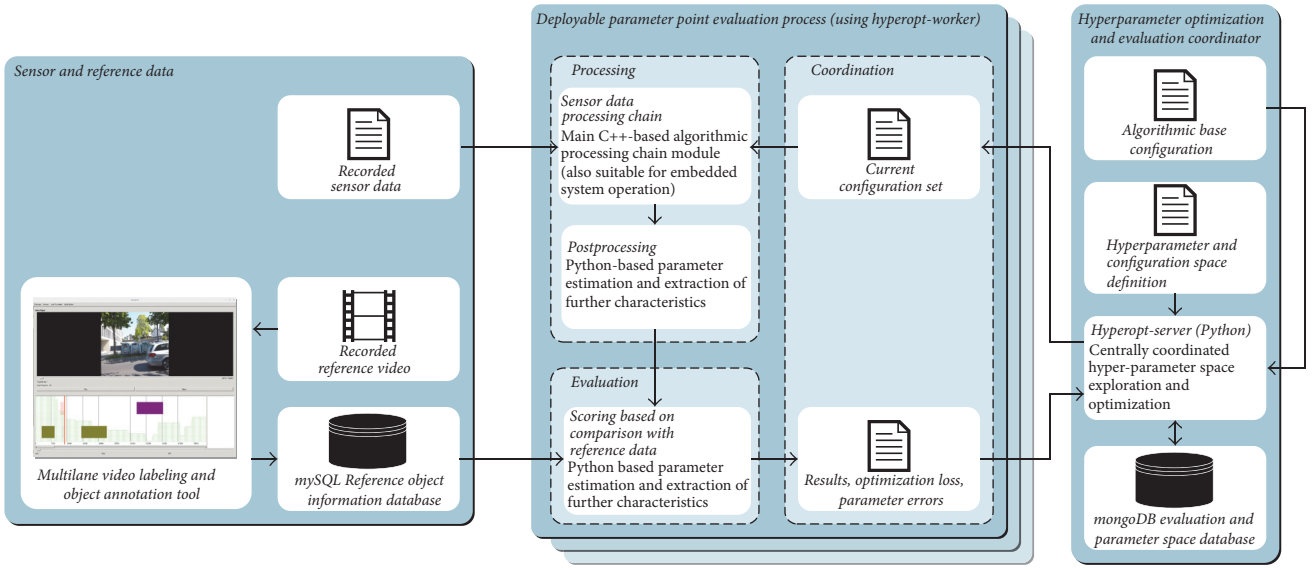


Figure 9: Evaluation framework and methodology for parameter space exploration.

5.1. Parameter Space Exploration and Optimization Methodology. In the following, the framework for parameter space exploration of the whole system with its corresponding algorithms is specified. The basic structure is shown in Figure 9.

Parameter Set Evaluation Concept. As a main principle, (hyper)parameter sets are given to a worker shown in the central block in the diagram. Each worker then evaluates a single parameter set (deployed with the *Coordination* subsystem) for real-world recorded sensor data using the C++-based main processing chain and Python-based postprocessing as part of the *Processing* subsystem. The processing principles were described in the implementation description of this article in Section 4.5. Then, in the *Evaluation* subsystem, the objects resulting from the processing stage are taken together with data from an object reference database. This allows the analysis of several performance metrics, such as the detection F_1 score and the correct lane assignment, which yields a performance score or optimization loss as a quality indicator of the parameter set. This is then reported back to the *Coordination* subsystem of the worker.

Reference Database. On the left side of the diagram, the sensor and reference data are shown. Reference object data is stored in a mySQL database. A reference video is recorded in parallel to a sensor data recording. This video is then manually processed using a labeling tool which is also shown in the diagram. The tool allows labeling traffic participant objects on multiple lanes in a timeline view and adding annotations to these objects such as the vehicle type, vehicle size, movement direction, and velocity, which is stored in the database. By analyzing the raw video material itself, the tool also provides the activity level in the video material which supports finding the next vehicle passing by in less frequented time ranges.

Cluster-Label-Pairing for Scoring. Both the resulting clusters from the processing of the raw sensor data and the objects in the reference database are events extended in the time dimension, with a specific start and end time. In the detection performance analysis of the system, each cluster (in case of perfect detection) would be paired with the corresponding reference label object. As the exact cluster-label-pair assignment is not the case in a real scenario, there can be ambiguities or false-positive/false-negative cases. Therefore, a bipartite cluster-label-graph is built up from the cluster and reference data, with edges between the nodes of specific cluster-label candidate pairs. Then, a maximum cardinality bipartite matching is performed. With the resulting cluster-label-pairs and the known sets of all reference and cluster objects, all scorings such as precision, recall, accuracy, and F_1-score can be calculated and reported back for the specific parameter set.

Coordination of (Hyper)parameter Sets. Given the possibility of using a worker process to evaluate parameter sets with the described procedure, the exploration of the (hyper)parameter space needs to be coordinated and optimized. This is achieved by using the hyperopt Python package [37], which allows the definition of several (hyper)parameters with their according distributions and properties and then to automatically explore this space using simulated annealing, random search, and tree of Parzen estimator techniques for optimization. The parameter sets and their according loss results are stored in a MongoDB database. New parameter sets are integrated into a base configuration file which is then assigned to the worker process. In order to speed up the optimization and training procedure, a large number of workers are deployed on an HPC. Typically, good convergence is achieved after 10,000 optimization iterations. If a faster convergence is desired, starting points for the parameters can be calculated with the investigations done in Section 4.3. Typical

Table 2: Overview of test scenarios and their characteristics.

Scenario name	Number of lanes	Sensor distance range[1]	Sensor mounting height	Typical speed range	Pulse interval T_{rep}	Description
Urban1	2	7.5 m	3.5 m	20–50 km/h	100 ms	Urban alley street with trees on both sides and parked cars on the adjacent side, mixed use by cars, buses and bicyclists, sensor placement in a horizontal distance of 1 m to curb
Urban2	2	8 m	3.5 m	20–40 km/h	100 ms	Sensor placement behind sidewalk, distance to curb 2 m, mixed use by cars, buses and bicyclists, reflective trash containers on adjacent side
Urban3	2	7 m	5 m	20–50 km/h	100 ms	Typical "urban canyon" with large buildings on both side close to the street and narrow sidewalks, distance to curb 20 cm
TestTrack	-	>15 m	3.5 m	5–100 km/h	50–100 ms	Automotive test track without reflective objects, only asphalted road surface in sensor range, low rate of vehicles passing by with highly different speeds

[1]The distance range is the distance from the ground projection point of the sensor position to the curb side or delimiter of the adjacent street side. For the calculation of the time-of-flight equivalent distance, both this sensor distance range and the mounting height have to be incorporated.

(hyper)parameters, which are part of the optimization process, are as follows:

(i) Dimensions of ε-environment (ε_T, ε_D) for the DBStaC algorithm and the core point threshold level MinSum (see Section 4.3.3)

(ii) Properties of the statistics memory, such as the statistics memory size (past information, see Section 4.4.2) and a storage subsampling factor

(iii) Optional use of statistical feedback (see Section 4.4.2) in the scenario for base distribution stabilization

(iv) Optional clipping threshold for calculated statistical norms in standardization to stabilize against outliers

(v) Optional use of matched filtering for the received envelope signal

5.2. Test Scenarios. In Table 2, the different test scenarios are given with their characteristics and description, covering a variety of urban environments and synthetic measurements on an automotive test track. The scenarios are identified by a specific name and will be referenced in the following, when performance results are given.

6. Results and Discussion

Results of the field tests and evaluations are given in this chapter. First, the according performance metrics are introduced and specified, followed by the results for the different scenarios and finally a discussion of the results.

6.1. Evaluation Metrics. In this paper, the main focus lies on the detection of traffic participants as objects and their correct assignment to the lanes of the street. For describing performance in our object detection scenario, no calculation of the true-negative count is possible, as we can have an arbitrary number of objects in our data timeline. Therefore, the widely used F_1 score is chosen as the main performance metric for detection, being the harmonic mean of precision and recall. The precision P is defined as the number of true positives (T_P) divided by all positive outcomes:

$$P = \frac{T_P}{T_P + F_P}. \tag{19}$$

In contrast, the recall R is defined as the number of true positives divided by the sum of true positive and false-negative (F_N) outcomes:

$$P = \frac{T_P}{T_P + F_N}. \tag{20}$$

Then, the F_1 score is defined as follows:

$$F_1 = 2\frac{P \cdot R}{P + R} = \frac{2 \cdot T_P}{2 \cdot T_P + F_N + F_P}. \tag{21}$$

A further advantage is that the combination of precision and recall into the F_1 score allows being independent of requirements biases, which can be a focus on either precision or recall performance, for example, when more false positives (F_P) or negatives are acceptable in a specific scenario. F_1 score calculation only requires the information of true positives, false positives, and false negatives. These numbers are calculated based on the bipartite cluster-label-graph assignment introduced in Section 5.1.

The second important metric is the assignment of objects to the correct lanes for determining the direction of the traffic flow. With multiple lanes as a multiclass problem, the F_1 score cannot be used directly. As an alternative, the weighted F_1 score is used, which first calculates the F_1 score for every lane (correct or incorrect assignment to specific lane) and then calculates a weighted mean of all lane scores, with the number of true reference object instances as the weight.

Table 3: Field test evaluation results for different scenarios.

Scenario name	Object count	F_1 score detection	Weighted F_1 score lane assignment
Urban1	224	0.98	0.91
Urban2	133	0.92	0.95
Urban3	305	0.97	0.98
TestTrack	17	0.94	–

6.2. Discussion of Test Results. The performance evaluation results for the different scenario with optimized parameters are shown in Table 3. Both detection and lane assignment scores are always larger than 0.9, being a very good performance level in real scenarios and satisfying the initially stated high requirements on traffic information quality. The single-sensor ultrasonic traffic participant detection is therefore possible with high performance without exploiting any directional information of the signal. This is facilitated by the algorithms proposed in this paper, which solved several prevailing problems in ultrasonic sensing techniques.

The multilane assignment performance is also very high and of special importance. As no directional or velocity information is available with these sensors, the known information of a fixed lane's driving direction yields the final traffic flow information with direction. A possible problem is showing up in the scenario *Urban1* with the lowest weighted F_1 score for assignment of 0.91: people were driving in the middle of the two-lane street several times, which compromised lane assignment performance.

Further future long-term analyses are required to analyze the long-term stability in scenarios with a highly fluctuating traffic flow and day-and-night cycle. However, with the system structure which relies on the standardized data together with the statistics memory, the sensitivity to long-term sensor drift and also production variations is limited. Further investigation is also required on the impact of the statistical feedback for stabilization of the statistical base information. As the parameter optimization is based on only a few parameters, the susceptibility to overfitting effects is limited in the shown results. Still, for future investigations, also the test performance with regard to future measurements in the same scenario is of high interest.

7. Conclusion and Future Work

In this paper, it was shown that sidefire ultrasonic sensing is a viable option for multilane traffic participant sensing, giving very good performance results in real-world urban scenarios with a single sensor. The functionality is enabled by a novel combination of standardization techniques based on windowed statistics and a modified density-based clustering algorithm, together called DBStaC. Several evaluations were performed, both in real urban environments and for special cases on an automotive test track. The proposed system is integrated into an evaluation and parameter optimization methodology, whose reference system is flexible to be extended with further object characteristics in future.

The ultrasonic sensor system deployed together with the street lamp platform was presented to be a Smart City technology suitable for high-coverage deployment in cities, enabling traffic monitoring and further applications. With this platform, distributed processing and sensor fusion can expand the possibilities of using available traffic participant information even more, for example, for future applications like trajectory prediction. Further future work can be the use of ultrasonic sensor arrays for acquiring directional and velocity information. In the field of algorithmic improvements, the investigation of more advanced hypothesis testing techniques and channel statistics analyses could yield further performance improvements. Also, the comparison of the presented algorithmic concept with state-of-the art supervised machine learning algorithms such as recurrent neural networks is planned.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. van Essen and A. van Grinsven, "Final report appendix 9: Interaction of GHG policy for transport with congestion and accessibility policies," Project Report, EU Transport GHG 2050, 2012.

[2] "Roadmap to a single european transport area - towards a competitive and resource efficient transport system," White Paper, European Commission, 2011.

[3] H. Mayer, "Air pollution in cities," *Atmospheric Environment*, vol. 33, no. 24-25, pp. 4029–4037, 1999.

[4] R. Arnott, A. de Palma, and R. Lindsey, "Does providing information to drivers reduce traffic congestion?" *Transportation Research Part A: General*, vol. 25, no. 5, pp. 309–318, 1991.

[5] A. Zanella, N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[6] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, 2011.

[7] S. Himmel, M. Ziefle, and K. Arning, *From Living Space to Urban Quarter: Acceptance of ICT Monitoring Solutions in an Ageing Society*, Springer, Berlin, Germany, 2013.

[8] W. Xue, L. Wang, and D. Wang, "A prototype integrated monitoring system for pavement and traffic based on an embedded sensing network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1380–1390, 2015.

[9] J. Gajda, R. Sroka, M. Stencel, A. Wajda, and T. Zeglen, "A vehicle classification based on inductive loop detectors," in *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics*, pp. 460–464, May 2001.

[10] L. Klein, D. Gibson, and M. Mills, *Traffic Detector Handbook*, vol. 1, no. FHWA-HRT-06-108, Federal Highway Administration, Turner-Fairbank Highway Research Center, 3rd edition, 2006.

[11] M. Hallenbeck and H. Weinblatt, *Equipment for Collecting Traffic Load Data*, Transportation Research Board, NCHRP Report 509, 2004.

[12] N. Garber and L. Hoel, *Traffic and Highway Engineering, SI Edition*, Cengage Learning, 2014.

[13] R. Möbus and U. Kolbe, "Multi-target multi-object tracking, sensor fusion of radar and infrared," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 732–737, IEEE, June 2004.

[14] I. Urazghildiiev, R. Ragnarsson, P. Ridderström et al., "Vehicle classification based on the radar measurement of height profiles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 245–253, 2007.

[15] I. Urazghildiiev, R. Ragnarsson, K. Wallin, A. Rydberg, P. Ridderstrom, and E. Ojefors, "A vehicle classification system based on microwave radar measurement of height profiles," in *Proceedings of the 2002 International Radar Conference*, pp. 409–413, Edinburgh, UK, October 2002.

[16] J. Tao, S. Chen, L. Yang, and Y. Hu, "Ultrasonic technique based on neural networks in vehicle modulation recognition," in *Proceedings of the The 7th International IEEE Conference on Intelligent Transportation Systems*, pp. 201–204, October 2004.

[17] E. U. Warriach and C. Claudel, "Poster abstract: A machine learning approach for vehicle classification using passive infrared and ultrasonic sensors," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks, IPSN 2013 - Part of CPSWeek 2013*, pp. 333-334, April 2013.

[18] T. Matsuo, Y. Kaneko, and M. Matano, "Introduction of intelligent vehicle detection sensors," in *Proceedings of the 1999 Intelligent Transportation Systems Conference*, pp. 709–713, Tokyo, Japan.

[19] H. Kim, J.-H. Lee, S.-W. Kim, J.-I. Ko, and D. Cho, "Ultrasonic Vehicle Detector for Side-Fire Implementation and Extensive Results Including Harsh Conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 3, pp. 127–134, 2001.

[20] V. Agarwal, N. V. Murali, and C. Chandramouli, "A cost-effective ultrasonic sensor-based driver-assistance system for congested traffic conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 486–498, 2009.

[21] C. Heipke, "Crowdsourcing geospatial data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 6, pp. 550–557, 2010.

[22] S. Hind and A. Gekker, "Outsmarting traffic, together: Driving as social navigation," *Exchanges: the Warwick Research Journal*, vol. 1, no. 2, pp. 165–180, 2014.

[23] M. B. Sinai, N. Partush, S. Yadid, and E. Yahav, "Exploiting social navigation," https://arxiv.org/abs/1410.0151v1.

[24] R. Bauza, J. Gozalvez, and J. Sanchez-Soriano, "Road traffic congestion detection through cooperative Vehicle-to-Vehicle communications," in *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks, LCN 2010*, pp. 606–612, October 2010.

[25] S. Jeon, E. Kwon, and I. Jung, "Traffic measurement on multiple drive lanes with wireless ultrasonic sensors," *Sensors*, vol. 14, no. 12, pp. 22891–22906, 2014.

[26] Y. Jo and I. Jung, "Analysis of vehicle detection with WSN-based ultrasonic sensors," *Sensors*, vol. 14, no. 8, pp. 14050–14069, 2014.

[27] H. Kuttruff, *Room Acoustics*, Spon Press, New York, NY, USA, 5th edition, 2009.

[28] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 1999.

[29] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[30] B. Gold, A. V. Oppenheim, and C. M. Rader, "Theory and implementation of the discrete Hilbert transform," *Symposium on Computer Processing in Communications*, vol. 19, 1970.

[31] M. Meissner, "Accuracy issues of discrete hubert transform in identification of instantaneous parameters of vibration signals," *Acta Physica Polonica A*, vol. 121, no. 1A, pp. A164–A167, 2012.

[32] L. Xu and T. Xu, *Digital Underwater Acoustic Communications*, Elsevier Science, 2016.

[33] S. C. Kumbhakar and C. A. K. Lovell, *Stochastic Frontier Analysis*, Cambridge University Press, Cambridge, UK, 2003.

[34] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, ser. KDD'96*, pp. 226–231, AAAI Press, 1996.

[35] J. Gan and Y. Tao, "DBSCAN revisited: Mis-claim, un-fixability, and approximation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD'15*, pp. 519–530, ACM, New York, NY, USA, 2015.

[36] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[37] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in Science Conference*, S. van der Walt, J. Millman, and K. Huff, Eds., 2013.

Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 6197549, 11 pages
https://doi.org/10.1155/2018/6197549

Research Article

Taxi Driver's Operation Behavior and Passengers' Demand Analysis Based on GPS Data

Xiaowei Hu, Shi An, and Jian Wang

School of Transportation Science and Engineering, Harbin Institute of Technology, No. 73, Huanghe River Road, Nangang District, Harbin, Heilongjiang Province 150090, China

Correspondence should be addressed to Xiaowei Hu; xiaowei_hu@hit.edu.cn

Received 8 August 2017; Revised 1 November 2017; Accepted 27 November 2017; Published 3 January 2018

Academic Editor: Anastasios Kouvelas

The existing research outputs paid less attention to the relationship between land use and passenger demand, while the taxi drivers' searching behavior for different lengths of observation period has not been explored. This paper is based on taxi GPS trajectories data from Shenzhen to explore taxi driver's operation behavior and passengers' demand. The taxi GPS trajectories data covers 204 hours in Shenzhen, China, which includes the taxi license number, time, longitude, latitude, speed, and whether passengers are in the taxi vehicle, to track the passenger's pick-up and drop-off information. This paper focuses on these important topics: exploring the taxi driver operation behavior by the measurements of activity space and the connection between different activity spaces for different time duration; mainly focusing on eight traffic analysis zones (TAZs) of Shenzhen and exploring the customer's real-time origin and destination demands on a spatial-temporal distribution on weekdays and weekends; taxi station optimization based on the passenger demand and expected customer waiting time distribution. This research can be helpful for taxi drivers to search for a new passenger and passengers to more easily find a taxi's location.

1. Introduction

Urban land use and built environment have been considered to affect residents' travel demand with three dimensions: design, density, and diversity [1]. Traffic engineers and urban planners have been paying more attention to explore the correlation between land use and transportation, including the land use influence on travel demand, the transport network impacts on the urban spatial development, and the integration of land use and transport system [2–6].

As an important mode, taxis play a key role in the urban passenger transportation market and provide a convenient and comfortable service for the passengers. In the taxi service study field, researchers usually adopt virtual customer origin-destination demand patterns to analyze the model [7–9], which is connected with the area land use situation, but cannot completely reflect the temporal and spatial characteristics of passenger demands. With the rapid development of Information and Communication Technologies (ICT), this provides more accurate access time and location information for the study of human mobility. Taxi vehicles equipped with Global Position System (GPS) can be served as city-wide probes, which can also provide the traffic condition, time, taxi speed, and location information, as well as whether there are passengers in the taxi. Based on the taxi GPS traces data, we can obtain the customers real-time origin and destination demand, which can assist researchers in validating the taxi service model [10–12].

As the GPS data on taxis are only recorded over a special period of time's passenger trajectory, it is difficult to analyze the human mobility over a longer period, but taxi GPS traces can be adopted to analyze urban transport and land use situations [13]. For the past few years, researchers have achieved more progress in this field, such as the researches of Li et al. (2011) [14], Zheng et al. (2011) [15], and Yue et al. (2012) [16].

Recently researchers have combined taxi GPS data with mathematical models (Lévy flights model or Zipf distribution law) to analyze the passenger's visiting frequency at one area [17], trip length distribution [18], and drivers' behavior [11, 19]. However, the existing researchers paid less attention to the taxi drivers' behavior for different lengths of observation

Table 1: An example trace of taxi GPS data, using the same vehicle as the example.

	Example	Data type	Remark
ID	粤B384R5	Character	Taxi vehicle ID
Date	2011-4-18	Date	
Time	0:00:33	Time	
Longitude	114.01810	Number	
Latitude	22.53283	Number	
Status	0	Boolean	0: no passenger; 1: have passenger
Velocity	13	Number	Instantaneous velocity: kilometers per hour (km/h)
Angle	5		Driving direction: 0, east; 1, southeast; 2, south; 3, southwest; 4, west; 5, northwest; 6, north; 7, northeast

period; meanwhile, the relationship between land use and passenger demand has not been explored.

So this paper focuses on the time series distribution dynamic characteristic of passenger's temporal variation in certain land use types and taxi driver's searching behavior connection between different activity spaces for different lengths of observation period. This paper focused on the following topics.

(1) Exploring the taxi driver operation behavior by the measurements of activity space and the connection between different activity spaces for different time duration

(2) Mainly focusing on eight TAZs of Shenzhen and exploring the customer's real-time origin and destination demand on spatial-temporal distribution on weekdays and weekends

(3) Taxi station optimization based on the passenger demand and expected customer waiting time distribution.

The structure of this paper is as follows. Section 2 reviews the urban land use and travel demand correlation, as well as taxi driver's searching behavior. In Section 3, we present the taxi GPS traces data source and analysis measurements in detail. Section 4 presents the results and discussions. Finally, we conclude this paper in Section 5.

2. Literature Review

Researchers usually use virtual customer origin-destination demand patterns to analyze the taxi service model, which can refer to Arnott (1996) [7], Yang and Wong (1998) [8], Wong et al. (2001) [20], Bian et al., (2007) [21], and Luo and Shi (2009) [9]. With the development of GPS hardware and communication technology, now we can collect taxi GPS traces data over longer periods than previous typical survey [16] and it also can provide more information in detail, such as trip length, travel time, and speed by time of day, which can assist researchers to validate the taxi service model. At present, some researchers also work on this field [22, 23]; Zhang and He (2011) [22] focused more on the spatial distribution of taxi services in one day, while Hu et al. (2011) [23] mainly analyzed the one-day taxi temporal distribution of customers' pick-up and drop-off times in Guangzhou, China.

Based on taxi GPS trace data, researchers can analyze urban transport and land use status at the macro level, which can cover the shortage of the traditional questionnaire survey [14–16, 24]. Yue et al. (2012) [16] calibrated the parameters of the spatial interaction models based on the taxi GPS traces data of the central business distinct in Wuhan. Liu et al. (2012) [25] explored the temporal patterns of urban-scale trip in Shanghai and found that urban land use and structure can be expressed by the taxi trip patterns.

Giraudo and Peruch (1988) [26] had divided the taxi operation into two phases, "the transport phase" and "the approach phase," which also can be used to represent the taxi with passenger and without passenger operation, respectively. The taxi driver's searching passenger behavior happens in "the approach phase." When the driver has dropped off the prior passenger, then he/she drives around the area or region searching for the next passenger after a short time.

For the taxi driver's individual characteristics (driving experience, road network familiarity, etc.) and randomness of the passenger's arriving, the driver's searching for the next passenger can be seen as a random variable. Luo (2009) [27] had expressed taxi driver's searching for the next passenger as a double exponential (Gumbel) distribution.

Liu et al. (2010) [11] described the taxi driver's operation patterns and difference between top drivers and ordinary drivers' behavior in Shenzhen and discussed taxi drivers' behavior based on the taxi daily GPS traces data; they analyzed the drivers' spatial selection behavior, operation behavior, and route choice behavior. But in the research of Liu et al. (2010) [11], they did not mention drivers' searching space behavior pattern.

This paper attempts to bridge these gaps between theoretical research and practical development, based on the taxi GPS trajectories data of Shenzhen to explore urban land use and taxi driver's operation behavior.

3. Data Source and Taxi Operation Activity Space Measurements

3.1. Data Source. In this research, we use the taxi GPS traces data of Shenzhen, China, which contains 3198 taxi records over nine consecutive days, from 18 April, 2011 (Monday), to the noon 26 April, 2011 (Tuesday), with a total of 204 hours. Table 1 shows the typical format of taxi trajectory data, including taxi location (longitude, latitude), speed, direction (angle), and passenger pick-up and drop-off information (status), with associated time information. The data collection time interval is generally around 5 to 15 seconds. Delays or missing data may occur depending on the

GPS signal, and additional records are collected when taxi load status changes.

Here we use R statistics software (R 2.13.2, from https://www.r-project.org/) and ArcGIS 9.3 (https://www.esri.com/zh-cn/arcgis/products/arcgis-pro/overview) software to handle taxi GPS data processing and statistics work.

The data analysis level and scope can be divided into three aspects, including the mean center analysis of the pick-up and drop-off events each day, passengers' spatial-temporal distribution of eight TAZs (traffic analysis zones) in the 204 continuous hours, and the taxi driver's searching behavior exploring from different level.

3.2. Taxi Operation Activity Space Measurements. The taxi operation activity measurements mainly are based on the basic parameters distributions, such as mean center, standard deviational ellipse, standard deviation of the X and Y coordinates, and kernel density. Based on existing researches, we divided these measurements into two categories, the spatial distribution category and the extended second moments of activity locations measurement category.

3.2.1. The Spatial Distribution Category. The mean center (MC) is the average location of taxi operation activity service events in the space (including the pick-up and drop-off event), which can be calculated by the following [28]:

$$(\dot{x}_{\mathrm{mc}}, \dot{y}_{\mathrm{mc}}) = \left[\frac{\sum_{j=1}^{n} x_j}{n}, \frac{\sum_{j=1}^{n} y_j}{n} \right], \tag{1}$$

where $\dot{x}_{\mathrm{mc}}$, $\dot{y}_{\mathrm{mc}}$ are the coordinates of the mean center, which can determine the space location of the MC; x_j, y_j are the coordinates of taxi service event j in two dimensions; n is the total number of taxi service events.

Standard deviational ellipse (SDE) [28, 29] can determine the directional factors of the spatial distribution and find the main direction of the taxi service event in space. The calculation of SDE can be expressed by θ (the SDE y-axis in the clockwise rotation angle); e_1,the major axis of SDE; and e_2, the minor axis of SDE. The detailed formulas are as follows:

$$\begin{aligned}
\theta &= \arctan \frac{\left[\sum \left(x_j - \dot{x}_{\mathrm{mc}}\right)^2 - \sum \left(y_j - \dot{y}_{\mathrm{mc}}\right)^2\right] + \sqrt{\left\{\left[\sum \left(x_j - \dot{x}_{\mathrm{mc}}\right)^2 - \sum \left(y_j - \dot{y}_{\mathrm{mc}}\right)^2\right]^2 + 4\left[\sum \left(x_j - \dot{x}_{\mathrm{mc}}\right)\left(y_j - \dot{y}_{\mathrm{mc}}\right)\right]^2\right\}}}{2\sum \left(x_j - \dot{x}_{\mathrm{mc}}\right)\left(y_j - \dot{y}_{\mathrm{mc}}\right)} \\
e_1 &= 2S_x = \sqrt{\frac{\sum \left[\left(x_j - \dot{x}_{\mathrm{mc}}\right)\cos\theta - \left(y_j - \dot{y}_{\mathrm{mc}}\right)\sin\theta\right]^2}{n-2}} \\
e_2 &= 2S_y = \sqrt{\frac{\sum \left[\left(x_j - \dot{x}_{\mathrm{mc}}\right)\sin\theta - \left(y_j - \dot{y}_{\mathrm{mc}}\right)\cos\theta\right]^2}{n-2}} \\
A &= \pi S_x S_y,
\end{aligned} \tag{2}$$

where A is area of SDE, S_x is the semimajor axis of SDE, S_y is the semiminor axis of SDE, and $\dot{x}_{\mathrm{mc}}$, $\dot{y}_{\mathrm{mc}}$ and x_j, y_j are consistent with formula (1).

3.2.2. The Extended Second Moments of Activity Locations Measurement Category. Each taxi driver's daily activity space area mean center may have the relationship with the centroid of the whole taxi drivers' activity space [19], similar to the Susilo & Kitamura (2005) [30] analysis of the worker's daily activity locations relationship. We can analyze taxi driver's day-to-day variation on activity space and statistically analyze the second moments of activity locations.

Figure 1 shows an illustration of the drop-off (pick-up) locations mean center of each taxi driver and all taxi drivers, which can analyze each taxi driver's day-to-day variation in the pick-up and drop-off activity space. Based on our statistics, the distance of the two MCs is mainly concentrated between 200 m and 400 m, which may reflect the taxi driver's searching behavior around a certain MC.

4. Analysis Results

4.1. Taxi Driver's Operation Behavior Analysis. In this section, we first explored the taxi driver operation behavior by the measurements of activity space and the connection between different activity spaces for different time duration. Here the MC and the locations mean center of each taxi driver and all taxi drivers have been used in the analysis. Figure 2 presents the spatial distribution of all taxi drivers' drop-off activity space mean center, which is analyzed by each day.

From Figure 2, we can find that taxi driver's drop-off activity space mean centers are mainly distributed around 22.562 to 22.576 (latitude) and 114.035–114.070 (longitude). And comparing the weekdays (from Monday to Friday) and weekends, there are two area distributions, which is from 1 a.m. to 6 p.m. and from 7 p.m. to 12 p.m., respectively. The red circle in Figure 2 shows the distribution from 7 p.m. to 12 p.m.

Figure 3 presents the spatial distribution of all taxi drivers' pick-up activity space mean center, which is analyzed by each day.

Table 2: The detailed information of the eight TAZs.

Number	TAZ name	Longitude	Latitude	Area/km^2	Land use and urban function
TAZ1	Futian CBD	114.055–114.065	22.530–22.540	2.0	CBD, Commercial center
TAZ2	Conventional financial center	114.105–114.123	22.534–22.548	0.9	Conventional financial center
TAZ3	Luohu Port	114.109–114.117	22.530–22.538	0.6	Port connecting with Hong Kong
TAZ4	Huanggang Port	114.068–114.080	22.518–22.529	1.0	Port connecting with Hong Kong
TAZ5	Huanggang Village	114.058–114.069	22.517–22.528	1.4	Communities are densely distributed
TAZ6	Huangbei Street	114.130–114.143	22.545–22.557	1.5	Residential area
TAZ7	Nanhu Street	114.113–114.131	22.539–22.545	1.3	Residential area
TAZ8	Dongmen Street	114.120–114.130	22.544–22.554	1.0	Commercial leisure and entertainment area

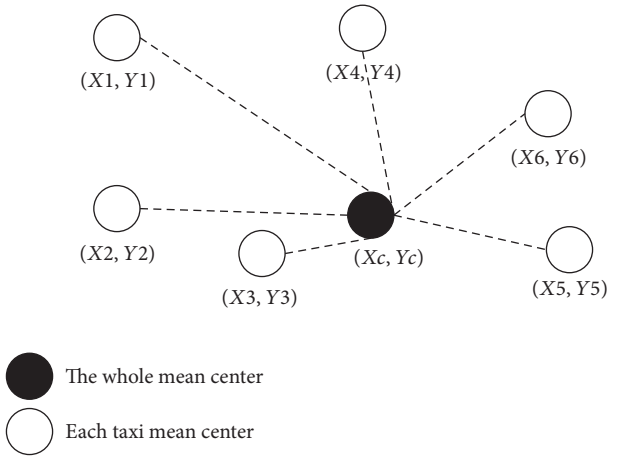


Figure 1: An illustration of the mean center locations of each taxi driver and all taxi drivers.

From Figure 3, we can also find that taxi driver's pick-up activity space mean centers are mainly distributed around 22.560 to 22.574 (latitude) and 114.035–114.070 (longitude). And comparing the weekdays (from Monday to Friday) and weekends, there are two area distributions, which is from 1 a.m. to 6 p.m. and from 7 p.m. to 12 p.m., respectively. The red circle in Figure 3 shows the distribution from 7 p.m. to 12 p.m.

From Figures 2 and 3, the scope of the taxi drivers' pick-up activity space mean center is more concentrated than the drop-off activity space mean center. Meanwhile, on the weekdays, the taxi drivers' pick-up activity space mean center and drop-off activity space mean center are more concentrated than on weekdays, which can reflect the passengers' daily life change, and they may have a more comfortable weekday.

4.2. Eight TAZs' Passenger Demands' Spatial-Temporal Distribution Analysis. In this section, we present the analysis results between passenger's origin and destination demand on spatial-temporal distribution from 18 April, 2011 (Monday), to the noon 26 April, 2011 (Tuesday). And we mainly focus on eight TAZs (see in Table 2) of Shenzhen; Figure 4 presents the eight TAZs' passenger pick-up (in blue line) and drop-off (in red line) statistical chart.

Based on Figure 4, there is a relative equilibrium of the total passenger pick-up and drop-off situation in the eight TAZs, but, for different land use types, there exist different peak hours of the pick-up and drop-off demand.

For the commercial and CBD areas (TAZ1, TAZ2), the passenger pick-up and drop-off service frequency is higher than in residential areas (TAZ6 and TAZ7). Meanwhile, the peak hour for the eight different land use areas is quite different from each other (see in Table 3).

By reason of different land use types, the peak hours of the eight TAZs are different from each other, while the passenger's pick-up and drop-off events are not synchronized. In Shenzhen, the peak hour of taxi passenger's is almost at the midnight, such as in TAZ2, TAZ7, and TAZ8, which is similar to the research of Hu et al. (2014).

The trend of how pick-up and drop-off changes with time is almost the same from Monday to Friday for each TAZ. At weekends, the peak hour is a bit different with in weekdays, especially in TAZ1, TAZ5, and TAZ6.

Then the taxi vehicle's service frequency for each TAZ was analyzed, which is shown in Table 4. From this table it can be seen that, in each TAZ, the taxi vehicle's supply is different to each other and each taxi vehicle's service time in TAZ is quite different. In Table 4, we can find that some taxi drivers are cruising around some areas, especially for the taxi drivers who provide more than 130 pick-up service in 204 hours (see in Table 4).

Based on this phenomenon, we divide the taxi drivers into different categories, some drivers only provide random service in the whole city, but some drivers can provide a relatively fixed service just around a specific area, such as the CBD, and residential area. Then the distributions of taxi drivers' pick-up service time in the eight TAZs were analyzed (as shown in Figure 5).

In TAZ1, TAZ5, and TAZ7, more than 60% of taxi driver's pick-up service times are less than 5 times, while, in TAZ3, TAZ4, TAZ6, and TAZ8, more than 85% of taxi driver's pick-up service times are less than 20 times, so 20 times can be taken as the boundary for the two different categories of taxi driver's service pattern. From Figure 5, we can also find that, in TAZ2, the average service time of each taxi driver is 46.47 times, and the 85% of taxi driver's pick-up service times is 70 times, so in TAZ2 the 70 times can serve as the boundary for the two different categories of taxi driver's service pattern.

4.3. Taxi Station Optimization. From the analysis, we can find that the biggest passenger demand is in TAZ2, which is along the Shenzhen south road and international trade

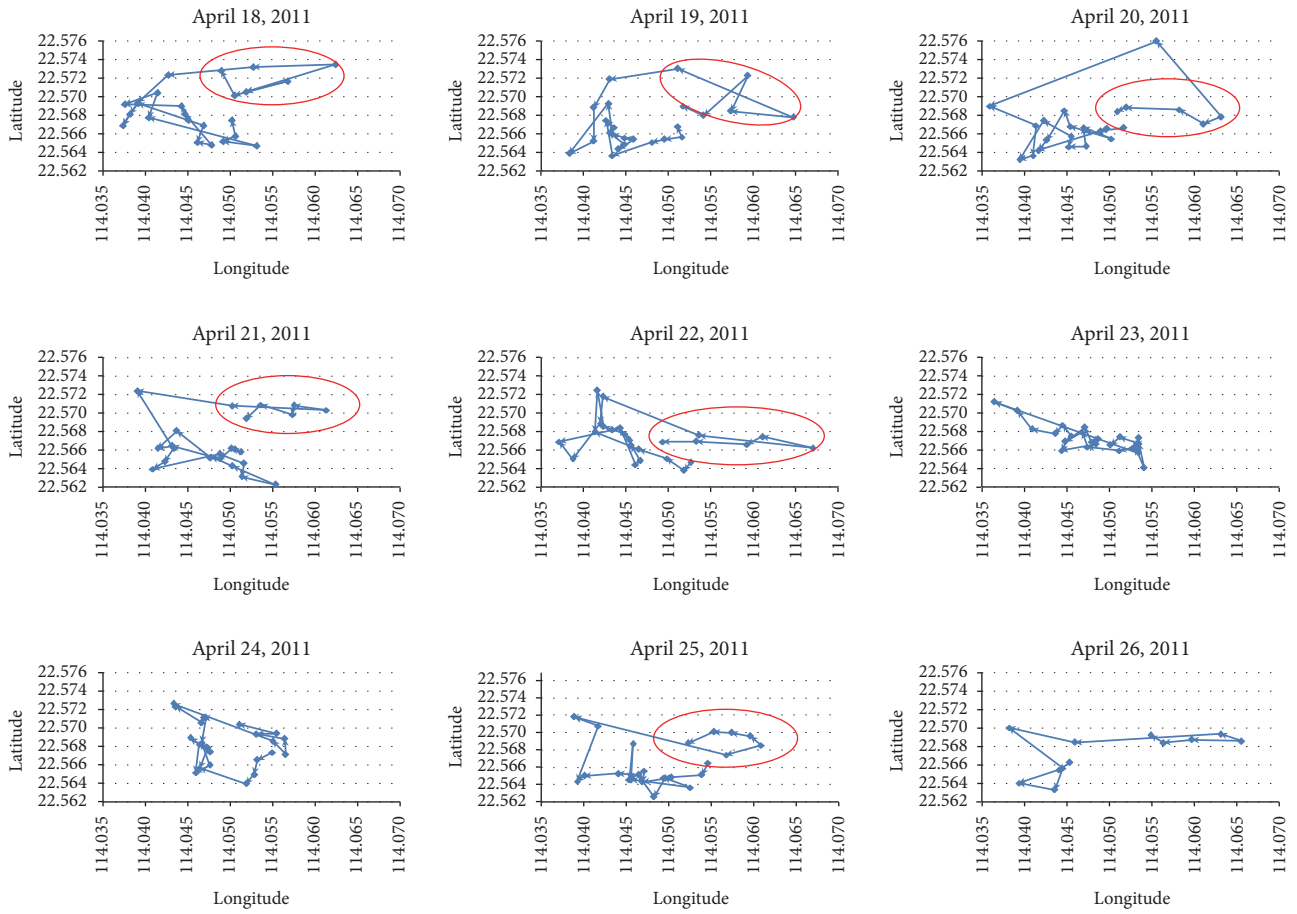


Figure 2: Spatial distribution of taxi driver's drop-off activity space mean center (from Monday to next Tuesday).

Table 3: Peak hour statistics of different TAZs.

Number	Activity	April 18 Monday	April 19 Tuesday	April 20 Wednesday	April 21 Thursday	April 22 Friday	April 23 Saturday	April 24 Sunday	April 25 Monday	April 26* Tuesday
TAZ1	Pick	12	14	9	15	10	23	22	15	9
	Drop	9	10	9	9	10	12	10	10	10
TAZ2	Pick	22	22	22	21	23	22	21	23	11
	Drop	22	22	16	21	23	22	21	15	12
TAZ3	Pick	23	23	16	21	24	24	24	15	10
	Drop	7	7	9	8	8	9	21	11	8
TAZ4	Pick	2	2	1	2	1	24	1	1	1
	Drop	8	8	8	8	23	1	23	1	8
TAZ5	Pick	10	20	1	22	1	20	23	1	10
	Drop	8	8	8	8	24	23	23	21	1
TAZ6	Pick	8	8	8	8	22	10	20	23	8
	Drop	22	23	23	22	22	22	21	23	1
TAZ7	Pick	22	22	1	23	23	2	2	23	1
	Drop	22	22	22	23	23	23	23	23	1
TAZ8	Pick	22	22	22	23	22	23	22	22	1
	Drop	22	22	23	23	22	21	21	21	1

* April 26, 2011, only has 12 hours of data recorded.

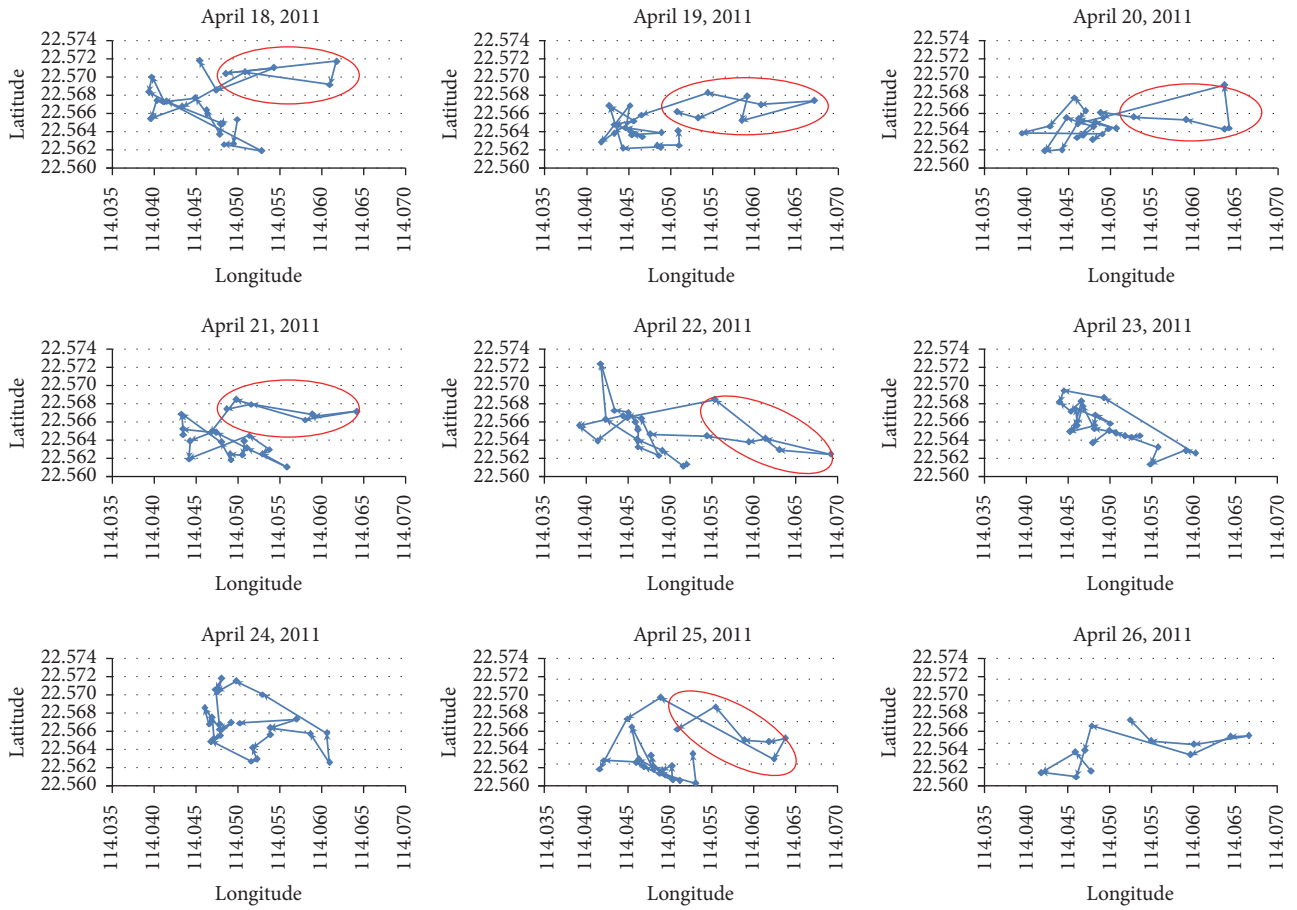


Figure 3: Spatial distribution of taxi driver's pick-up activity space mean center (from Monday to next Tuesday).

Table 4: Summary of each taxi vehicle's service frequency of each TAZ.

	TAZ1	TAZ2	TAZ3	TAZ4	TAZ5	TAZ6	TAZ7	TAZ8
Minimum	1	1	1	1	1	1	1	1
1st quarter	4	28	8	1	7	2	8	6
Median	6	44	14	3	12	4	15	10
Mean (average value)	7.392	46.47	14.92	4.163	15.35	5.737	18.02	12.65
3rd quarter	9	62	20	5	20	7	23	17
Maximum	76	**185**	84	34	**130**	85	**130**	109
Number of taxi vehicles	2,468	2,498	2,467	1,985	2,478	2,207	2,465	2,436
Total pick-up service times	18,244	116,072	36,812	8,264	38,044	12,661	44,414	30,811

center; at present this TAZ does not have taxi service station, which is inconvenient for passenger's travel, so this TAZ area needs to consider optimizing the taxi service station.

From Figure 4, we can find the two peak hours of passengers' pick-up service in TAZ2 is 2 p.m. to 3 p.m. and 9 p.m. to 10 p.m., which is connected with the land use and geographic location. So the taxi station optimization is based on the passenger demand and expected customer waiting time distribution, while we do not consider the setting form of the taxi station in this paper.

For the study field of taxi station's service area, Daganzo (1978) [24] proposed the flexible transit design model (FTDM), and in 2012 he had optimized it into a transit optimization approach [31]. Based on existing research of Nourbakhsh and Ouyang (2012) [32] and Sathaye (2014) [33], here a taxi station optimization model is presented to determine the service radius R.

According to the research of Nourbakhsh and Ouyang (2012) [32], each passenger's expected walk distance is shown in the following formula in km:

$$E = \frac{2D}{3}, \tag{3}$$

where D is the length of the side of one square; then each passenger's expected walk time in hours is

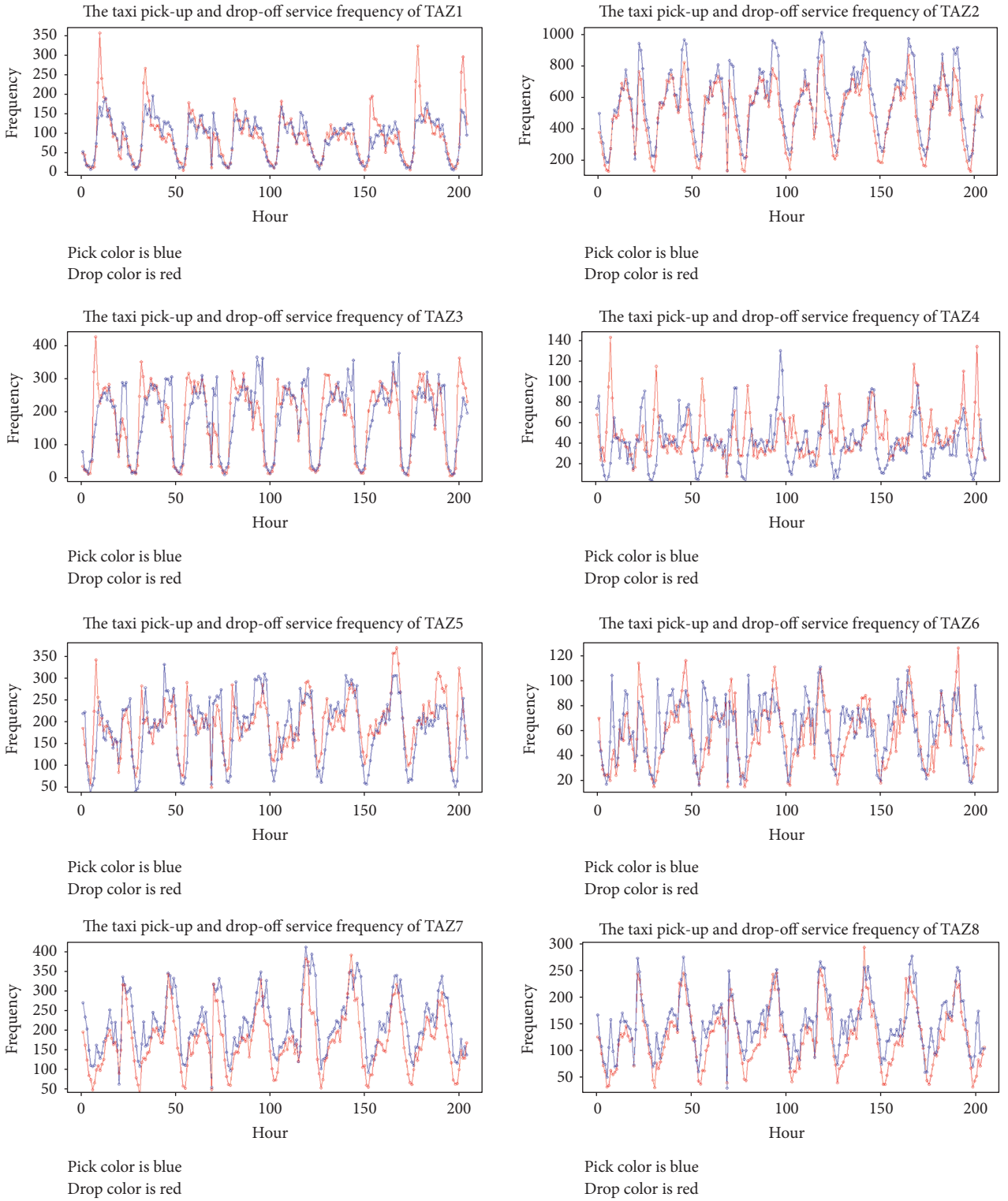


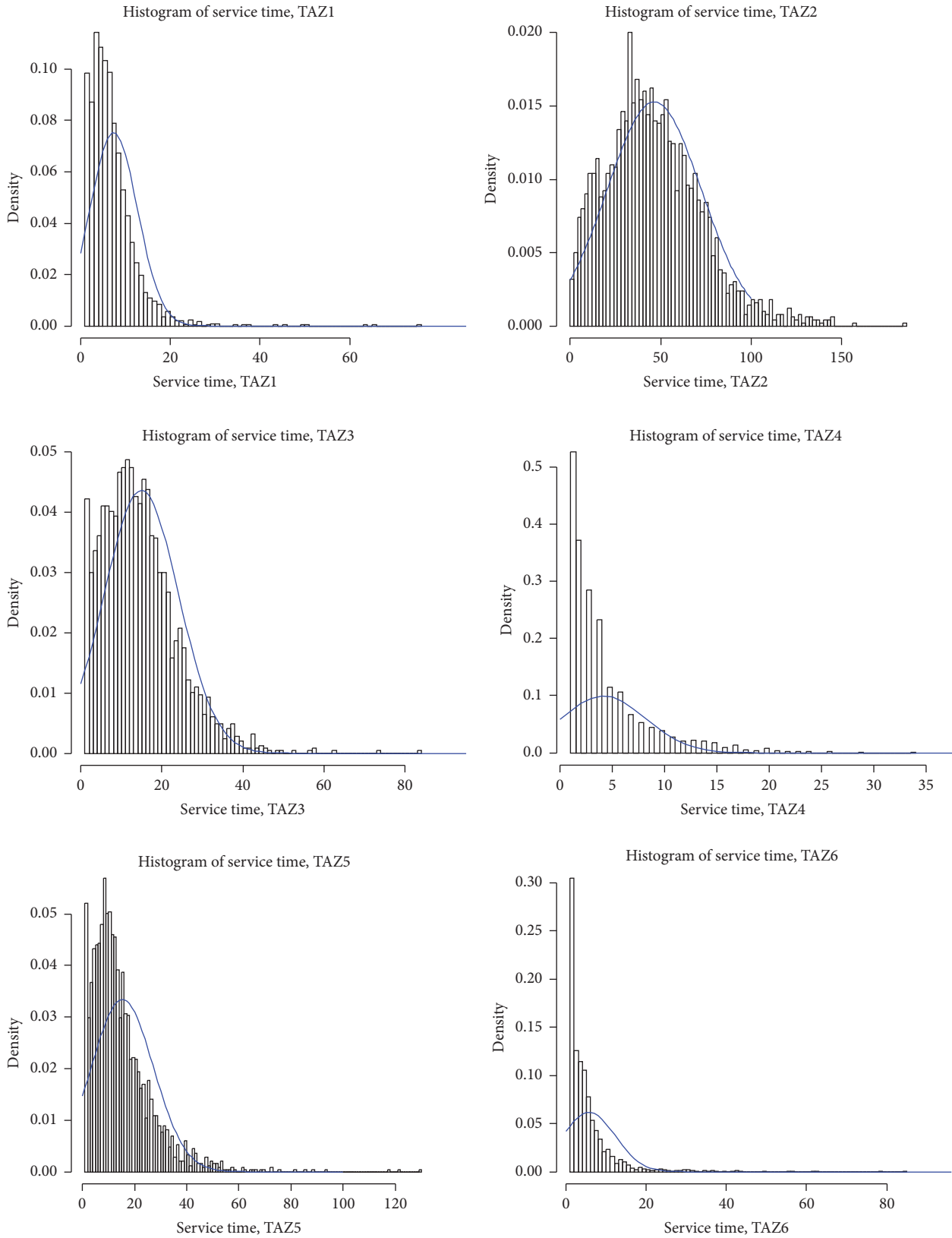
Figure 4: Eight TAZs passenger pick-up (in blue line) and drop-off (in red line) service statistics.

$$T = \frac{E}{v} = \frac{2D}{3v}, \tag{4}$$

where v is the average operation speed (km/h). Therefore, a taxi station's service radius R can be expressed by the following formula:

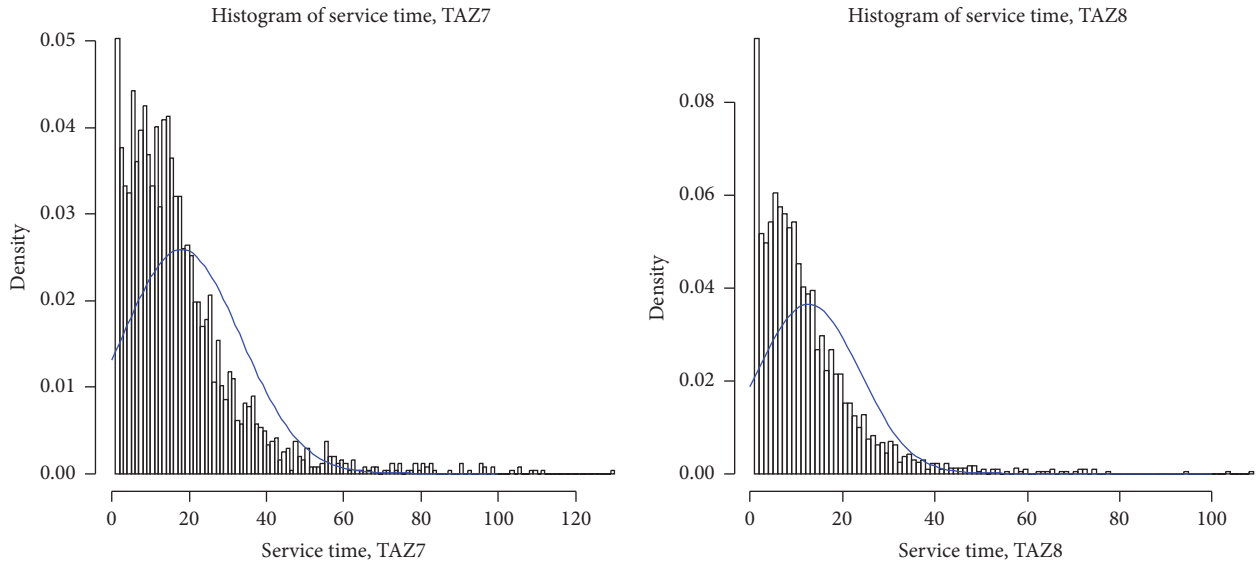
$$R = \min\left(\frac{D}{2}, \frac{0.95D}{\sqrt{\pi Y}}\right), \tag{5}$$

where R is service radius of taxi station (km) and Y is the number of taxi stations.



(a)

Figure 5: Continued.



(b)

Figure 5: The taxi vehicles' pick-up service frequency statistics of each TAZ in Shenzhen.

Table 5: Calculation table of taxi station's service radius (unit: km).

D (kilometer)	$Y = 1$	$Y = 2$	$Y = 3$	$Y = 4$
0.2	0.1	0.075799	0.061890	0.053598
0.4	0.2	0.151598	0.123779	0.107196
0.6	**0.3**	0.227397	0.185669	0.160794
0.8	0.4	**0.303196**	0.247559	0.214392
1	0.5	0.378995	**0.309448**	0.267990
2	1	0.757990	0.618897	0.535980

For the given D and Y, we can calculate the taxi station's service radius; the results are shown in Table 5. Referring to the study by Zhang et al. (2015) [34], which is based on taxi GPS data and analysis, they recommend the taxi station's service distance to be 300 m; this result can be matched with some results in Table 5 (the bold result).

5. Conclusions and Recommendations

This paper is based on taxi vehicle's GPS data to analyze the time series distribution dynamic characteristics of passengers' temporal variation in certain land use types and taxi driver's searching behavior in connection with different activity spaces for different lengths of observation period. And adopting GPS data had identified the passengers' demand hot area and proposed a taxi station optimization model, which can be served as reference to taxi station location decision.

By researching and proposing appropriate measures and statistics to properly measure and analyze activity spaces while recognizing their geographical dependence, this study can make some contributions to methodologies in measuring and analyzing behavioral dynamics. By understanding the dynamics in the activity spaces of taxi drivers over time, this study directly contributes to the field of travel behavior dynamics. The value that the study will potentially render in policy guidance also cannot be underestimated, understanding these dynamics at the driver level. The analysis will also provide a new method to optimize urban transport management, to investigate land-using planning, and to evaluate road network traffic conditions.

However, this paper from taxi vehicle's GPS data can reflect driver's behavior more accurately and, regarding the passenger level, it needs to combine the passenger's characters survey and the booking data from an Internet-booking application with the taxi vehicle's GPS data [35, 36] and to analyze passenger's trip and the relationship with land use. In addition, taxi service and the public passenger transport system are strongly complementarity in big cities. In the future, we will take into account the main public transit facilities on taxi demand analysis.

On the other hand, based on the expected customer waiting time distribution, the approximate distribution of the customer waiting time formula can be validated and modified in different kinds of land use type, which will be more useful in describing the principal characteristics in the taxi market and affect customers' decision and taxi driver's cruising time [37, 38].

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (71603063), the Natural Science Foundation of Heilongjiang (no. E2016032), China Postdoctoral Science Foundation funded Project (no. 2013M540299), and the Fundamental Research Funds for the Central Universities (no. HIT.NSRIF.2015075). The authors would like to express their sincere gratitude to Professor Xuesong Zhou (Arizona State University) for his valuable suggestions.

References

[1] R. Cervero and K. Kockelman, "Travel demand and the 3Ds: density, diversity, and design," *Transportation Research Part D: Transport and Environment*, vol. 2, no. 3, pp. 199–219, 1997.

[2] T. J. Kim, "A combined land use-transportation model when zonal travel demand is endogenously determined," *Transportation Research Part B: Methodological*, vol. 17, no. 6, pp. 449–462, 1983.

[3] K. T. Geurs, B. Van Wee, and P. Rietveld, "Accessibility appraisal of integrated land-use-transport strategies: methodology and case study for the Netherlands Randstad area," *Environment and Planning B: Planning and Design*, vol. 33, no. 5, pp. 639–660, 2006.

[4] J. Ying, "Continuous optimization method for integrated land use/transportation models," *Journal of Transportation Systems Engineering and Information Technology*, vol. 7, no. 3, pp. 64–72, 2007.

[5] P. Waddell, G. F. Ulfarsson, J. P. Franklin, and J. Lobb, "Incorporating land use in metropolitan transportation planning," *Transportation Research Part A: Policy and Practice*, vol. 41, no. 5, pp. 382–410, 2007.

[6] M. Aljoufie, M. Zuidgeest, M. Brussel, and M. van Maarseveen, "Spatial-temporal analysis of urban growth and transportation in Jeddah City, Saudi Arabia," *Cities*, vol. 31, pp. 57–68, 2013.

[7] R. Arnott, "Taxi travel should be subsidized," *Journal of Urban Economics*, vol. 40, no. 3, pp. 316–333, 1996.

[8] H. Yang and S. C. Wong, "A network model of urban taxi services," *Transportation Research Part B: Methodological*, vol. 32, no. 4, pp. 235–246, 1998.

[9] D. Luo and F. Shi, "A taxi service network equilibrium model with the influenced of demand distribution," *Journal of Railway Science and Engineering*, vol. 6, no. 1, pp. 87–91, 2009.

[10] C. F. Daganzo, "Urban gridlock: macroscopic modeling and mitigation approaches," *Transportation Research Part B: Methodological*, vol. 41, no. 1, pp. 49–62, 2007.

[11] L. Liu, C. Andris, and C. Ratti, "Uncovering cabdrivers' behavior patterns from their digital traces," *Computers, Environment and Urban Systems*, vol. 34, no. 6, pp. 541–548, 2010.

[12] C. Kang, X. Ma, D. Tong, and Y. Liu, "Intra-urban human mobility patterns: an urban morphology perspective," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 4, pp. 1702–1717, 2012.

[13] G. Mintsis, S. Basbas, P. Papaioannou, C. Taxiltaris, and I. N. Tziavos, "Applications of GPS technology in the land transportation system," *European Journal of Operational Research*, vol. 152, no. 2, pp. 399–409, 2004.

[14] Q. Li, T. Zhang, H. Wang, and Z. Zeng, "Dynamic accessibility mapping using floating car data: a network-constrained density estimation approach," *Journal of Transport Geography*, vol. 19, no. 3, pp. 379–393, 2011.

[15] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*, pp. 89–98, ACM, Beijing, China, September 2011.

[16] Y. Yue, H. Wang, B. Hu, Q. Li, Y. Li, and A. G. O. Yeh, "Exploratory calibration of a spatial interaction model using taxi GPS trajectories," *Computers, Environment and Urban Systems*, vol. 36, no. 2, pp. 140–153, 2012.

[17] B. Jiang, J. Yin, and S. Zhao, "Characterizing the human mobility pattern in a large street network," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, no. 2, Article ID 021136, 2009.

[18] X. Hu and J. Feng, "Research on characteristics of taxi traffic based on GPS data," *Urban Transport of China*, vol. 5, no. 2, pp. 91–95, 2007 (Chinese).

[19] X. Hu, S. An, and J. Wang, "Exploring urban taxi drivers' activity distribution based on GPS data," *Mathematical Problems in Engineering*, vol. 2014, Article ID 708482, 2014.

[20] K. I. Wong, S. C. Wong, and H. Yang, "Modeling urban taxi services in congested road networks with elastic demand," *Transportation Research Part B: Methodological*, vol. 35, no. 9, pp. 819–842, 2001.

[21] Y. Bian, W. Wang, and J. Lu, "Equilibrium model of urban taxi service network," *Journal of Traffic and Transportation Engineering*, vol. 7, no. 1, pp. 93–98, 2007.

[22] Z. Zhang and X. He, "Analysis and application of spatial distribution of taxi service in city subareas based on taxi GPS data," in *Proceedings of the 11th International Conference of Chinese Transportation Professionals: Towards Sustainable Transportation Systems (ICCTP '11)*, pp. 1232–1243, Nanjing, China, August 2011.

[23] J.-H. Hu, H.-Y. Xie, and G.-P. Zhong, "A method for determining taxi reasonable scale based on floating car data," in *Proceedings of the 1st International Conference on Transportation Information and Safety: Multimodal Approach to Sustained Transportation System Development - Information, Technology, Implementation, ICTIS 2011*, pp. 1255–1262, China, July 2011.

[24] C. F. Daganzo, "An approximate analytic model of many-to-many demand responsive transportation systems," *Transportation Research*, vol. 12, no. 5, pp. 325–333, 1978.

[25] Y. Liu, F. Wang, Y. Xiao, and S. Gao, "Urban land uses and traffic 'source-sink areas': evidence from GPS-enabled taxi data in Shanghai," *Landscape and Urban Planning*, vol. 106, no. 1, pp. 73–87, 2012.

[26] M.-D. Giraudo and P. Peruch, "Spatio-temporal aspects of the mental representation of urban space," *Journal of Environmental Psychology*, vol. 8, no. 1, pp. 9–17, 1988.

[27] D. Luo, *Urban Mixed Traffic Network Equilibrium Analysis under the Influence of Taxi Services*, Dissertation of Central South University, Changsha, China, 2009.

[28] J. Lee and D. W. S. Wong, *Tatistical Analysis with Arcview GIS*, John Wiley& Sons Inc, New York, NY, USA.

[29] Ned Levine & Associates, CrimeStat: A Spatial Statistics Program for the Analysis of Crime Incident Locations (v 3.0). Houston, TX, and the National Institute of Justice, Washington, DC, USA, 2014.

[30] Y. O. Susilo and R. Kitamura, "Analysis of the day-to-day variability in the individual's action space: an exploration of the six-week Mobidrive travel diary data," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1902, pp. 124–133, 2005.

[31] C. F. Daganzo, "On the design of public infrastructure systems with elastic demand," *Transportation Research Part B: Methodological*, vol. 46, no. 9, pp. 1288–1293, 2012.

[32] S. M. Nourbakhsh and Y. Ouyang, "A structured flexible transit system for low demand areas," *Transportation Research Part B: Methodological*, vol. 46, no. 1, pp. 204–216, 2012.

[33] N. Sathaye, "The optimal design and cost implications of electric vehicle taxi systems," *Transportation Research Part B: Methodological*, vol. 67, pp. 264–283, 2014.

[34] C. Zhang, S. Zhang, and M. Wang, "Location selection method for taxi stands based on GPS data," *Transport Research*, vol. 1, no. 4, pp. 42–48, 2015.

[35] S. M. Zoepf and D. R. Keith, "User decision-making and technology choices in the U.S. carsharing market," *Transport Policy*, vol. 51, pp. 150–157, 2016.

[36] S. Harding, M. Kandlikar, and S. Gulati, "Taxi apps, regulation, and the market for taxi journeys," *Transportation Research Part A: Policy and Practice*, vol. 88, pp. 15–25, 2016.

[37] S. De Vany, "Capacity utilization under alternative regulatory restraints: an analysis of taxi markets," *Journal of Political Economy*, vol. 83, no. 1, pp. 83–94, 1975.

[38] H. Yang, S. C. Wong, and K. I. Wong, "Demand-supply equilibrium of taxi services in a network under competition and regulation," *Transportation Research Part B: Methodological*, vol. 36, no. 9, pp. 799–819, 2002.

Hindawi
Journal of Advanced Transportation
Volume 2017, Article ID 8483750, 16 pages
https://doi.org/10.1155/2017/8483750

Research Article

Improvement of Network Performance by In-Vehicle Routing Using Floating Car Data

Gerdien A. Klunder,[1] Henk Taale,[2] Leon Kester,[3] and Serge Hoogendoorn[2]

[1]*TNO, Sustainable Urban Mobility and Safety, The Hague, Netherlands*
[2]*Department of Transport & Planning, Delft University of Technology, Delft, Netherlands*
[3]*TNO, The Hague, Netherlands*

Correspondence should be addressed to Gerdien A. Klunder; gerdien.klunder@tno.nl

Received 3 July 2017; Accepted 26 October 2017; Published 5 December 2017

Academic Editor: Anastasios Kouvelas

This paper describes a study which gives insight into the size of improvement that is possible with individual in-car routing advice based on the actual traffic situation derived from floating car data (FCD). It also gives an idea about the required penetration rate of floating car data needed to achieve a certain degree of improvement. The study uses real loop detector data from the region of Amsterdam collected for over a year, a route generating algorithm for in-car routing advice, and emulated floating car data to generate the routing advice. The case with in-car routing advice has been compared to the base case, where drivers base their routing decisions on average knowledge of travel times in the network. The improvement in total delay using the in-vehicle system is dependent on penetration rate and accuracy of the floating car data and varies from 2.0% to 3.4% for 10% penetration rate. This leads to yearly savings of about 15 million euros if delay is monetarised using standard prices for value of time (VOT).

1. Introduction

By routing individual vehicles, it is clear that individual travel times and possibly also total network travel time can be improved. With recent technologies, a personal routing advice can be determined and presented to individual car drivers, by using an in-car device. This routing advice can be based on real-time traffic data, for example, floating car data generated by other drivers using the same service, which comprises experienced travel times. Recently, several pilots within the "Practical Trial Amsterdam" have been performed to test such a service [1, 2] in the Netherlands. However, relatively small implementation and compliance rates gave little insight into the potential effects for large-scale implementation, since route choice effects may positively influence travel times due to better utilization of available capacity, but only when a sufficient number of drivers will change their route.

In order to be able to determine a good routing advice based on the current traffic situation, adequate knowledge of this situation is necessary. Especially in the case of nonrecurrent, unexpected events, such as car accidents, routing advice can be very useful to shorten travel times. However, in order to detect such an unexpected event, sufficient and real-time traffic measurements need to be available. On Dutch motorways traffic measurements of loop detectors are available and of good quality, but much less traffic measurements are available for urban networks. For those networks floating car data can be used as an additional source. However, this raises certain questions. For example, what should be the amount and quality of FCD in order to be able to determine an adequate routing advice? And, what may be the improvement of such routing advice, both for the individual driver and for the network as a whole? This paper tries to answer these questions. First the research approach is presented; then the importance of determining the relation between quality of traffic data and the performance of a traffic management measure is explained. Next, the underlying data and the smart routing algorithm are described, after which the results are presented. Finally, the research questions are answered, conclusions are drawn, and recommendations for further research are given.

2. Importance of Determining the Relation between Quality of Traffic Data and the Performance of Traffic Management

Traffic management measures are designed with the (underlying) assumption that one has perfect traffic data available, which the traffic management system uses to operate. However, since in reality traffic data are never perfect, the traffic management measure will not perform optimally. But what effect can be achieved with these inaccurate or incomplete data and how much it differs from the "optimal" situation are usually not known. If these were known, it is, for example, possible to do a cost-benefit analysis on the goals the measure will achieve in relation to the costs of the data collection. Is it worth equipping more measuring points or more people with a measuring device? For ramp metering a start has been made to estimate the impact of inaccurate data and the benefits if cameras are used instead of loop detectors [3].

In our specific case, it is already interesting to find out how a measure as smart routing will perform in an urban network anyway, where most of the people already have some knowledge of the (regular) congestion. Another question could be how many data should be gathered (by floating devices) in order to obtain at least a positive effect. Finally, the effectiveness of route advice also depends for a large part on the normal route choice behaviour, the degree of succession of the advices, and access to other types of traffic information that the users already have. For this study these aspects are not taken into account, because this would require large-scale research on driving behaviour. However, the results can be interpreted for lower degrees of succession by looking at the results for lower penetration rates of the system.

3. Research Approach

To answer the research questions about the relation between the accuracy of data and the efficiency of the routing advice, a combination of modelling and using real (historical) data measurements has been chosen. The ground truth is derived from real traffic data measurements, while the quality variations of the FCD are modelled as perturbations of the real traffic data. Furthermore, since driven routes were not available (as is often the case), route choices have been modelled with commonly used mathematical models such as the logit model, as will be explained later. For a selected day, the historical speeds are used as input, while different situations for different FCD qualities and penetration rates of the smart routing system are modelled. To show the effect of different route choice options, several performance indicators are calculated on network level, in order to be able to quantify the effect of the differences in FCD quality.

Several modelling and data processing steps have been taken. The steps and the relation between them are shown in Figure 1. The first step was to know the ground truth of the traffic situation in the network. This means that the real speeds in the network during the investigation period should be available from measurements or estimations derived from measurements. For this, a large database with traffic measurements in the Netherlands was used. This traffic database was collected by the Dutch National Data Warehouse for Traffic Information (NDW, see [4]) and consists mainly of loop detector data. From this database the information for the region of Amsterdam was derived. A ground truth was created by using all available traffic data and using gap filling methods to complete missing data, as explained later.

The second step was to define zones in the network and to derive an origin-destination (OD) matrix with the number of trips from each origin to each destination for a given time period.

The third step was to determine reasonable alternative routes in the network between each OD pair, in order to be able to distribute the traffic over the network for the base situation and for the situation with smart routing. For each OD pair, a number of alternative routes were determined, based on aspects such as travel time, trip length, and road type.

These first three steps were preprocessing steps, independent of the smart routing system. The following steps were needed to test the smart routing measure for a selected day and for all variations of the FCD quality, time of day, and different penetration rates of the system.

Therefore the fourth step was the route selection, both for the normal users and for the users using smart routing. For the base situation of this study, without routing advice, a multinomial logit model was used to distribute the drivers over the route alternatives, based on the average network speeds. Link travel times were updated using calibrated BPR functions [5]. For the alternative situation using smart routing, the drivers were distributed either over only the shortest route or over three route alternatives with the shortest travel time (out of the predetermined set with route alternatives), using actual travel times and link speeds, as will be explained in more detail later. The network improvement was then determined for various penetration rates of the system and various times of the day and days of the year.

In order to determine the effect of the quality of the FCD, the amount of available FCD was varied by drawing a random number of links for which an actual speed measurement is assumed available. Links with a higher flow have a larger probability to be selected, since the probability that an FCD vehicle is found on these links is higher. This is accomplished by drawing a weighted sample with weights equal to the flow. For the links that are not covered by this data sample, the average (historical) link speed is used. Furthermore, in order to investigate the effect of inaccurate speed measurements, the quality of the floating car data was varied by adapting the link speeds with a random error. Both cases (variations in the amount and quality of FCD) were tested separately.

Finally, for varying implementation rates of the system, the improvement in travel times and delays (both individually and network-wide) was determined with regard to the base case and with regard to the optimal case based on perfect information.

4. Underlying Data

For the case study, we used different types of traffic data, such as loop detector data, travel times measured with cameras, and FCD, originating from the NDW database and

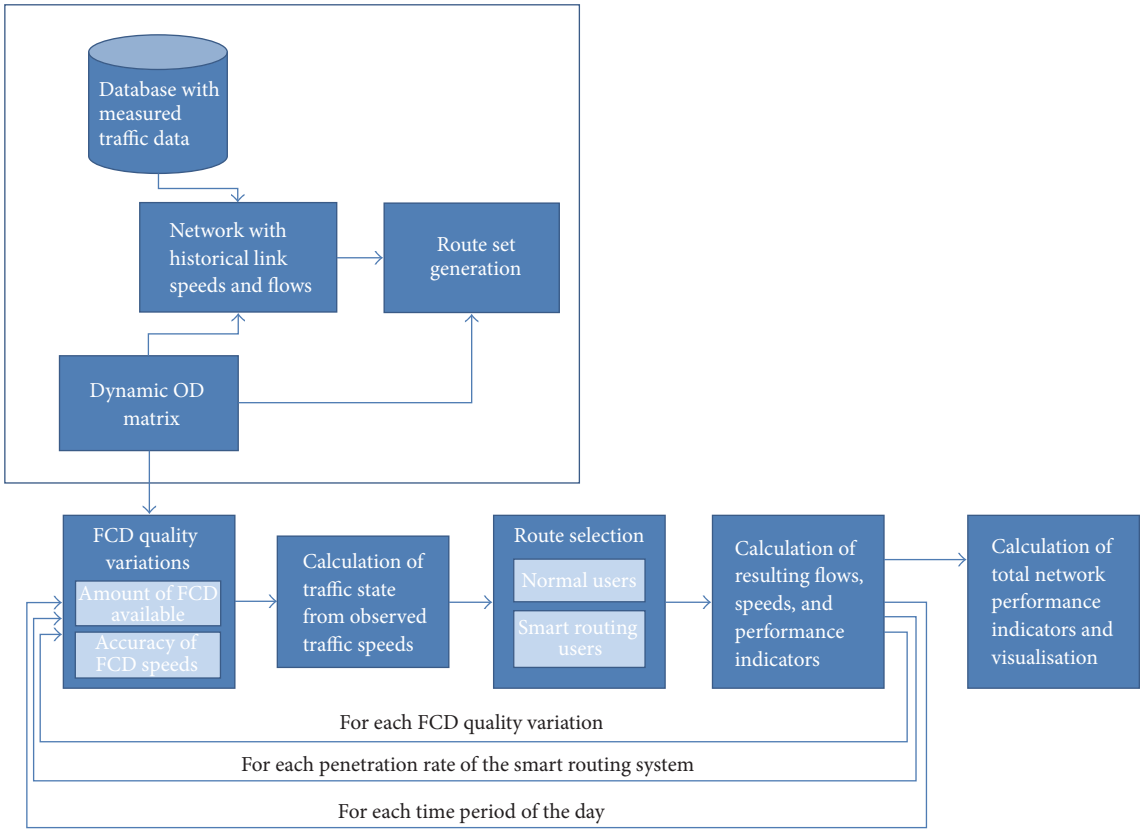


Figure 1: Modelling approach for the smart routing case study.

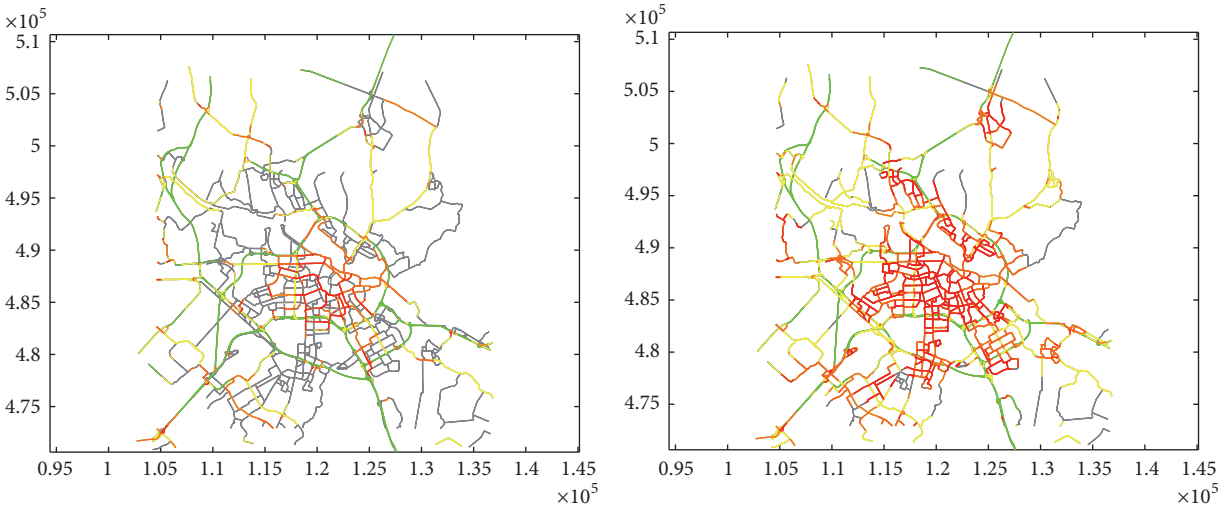


Figure 2: Modelled network of the region of Amsterdam, before (a) and after (b) gap filling. The colours indicate the average speed: green > 80 km/h, yellow 50–80 km/h, orange 30–50 km/h, and red 0–30 km/h. For the grey links, no speed data could be derived.

the Practical Trial Amsterdam [6], fused into a database developed by TNO.

The network includes the city centre of Amsterdam and the surrounding region with a diameter of about 30 kilometres, as shown in Figure 2. It consists of 12,425 links. After gap filling and filtering, speeds and flows are available for most of the links for every minute of the day. For links for which no historical information was available, the gap filling method determines a link speed based on the available speed information of all links of the same road type within a range of 2 kilometres, with weights inversely proportional to the distance.

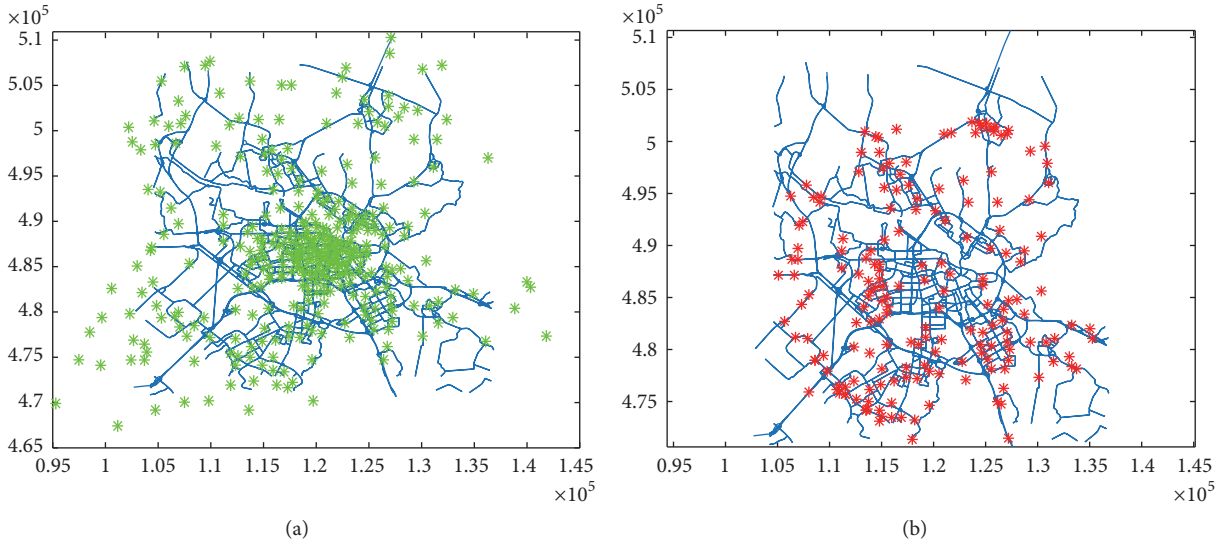


Figure 3: Zone aggregation in the network, (a) 350 zones before aggregation and (b) 183 zones after aggregation.

An OD matrix was obtained from a modelling study with the Regional Traffic Management Explorer [7]. This matrix came from a calibrated strategic model and was made dynamic by using departure time profiles, derived from inquiries among travellers. The matrix consisted of 350 zones with a high zonal density in the city centre, which is rather detailed for this small region. It turned out that this level of detail was not practical for this case study in terms of long calculation times and relevance for route choice. Therefore, the original matrix was aggregated into a smaller matrix of 183 zones. The (aggregated) zones also needed to be connected to the network (partly manually). The original zones (centres) and the aggregated zones are shown in Figure 3. A stronger aggregation was done within the city centre, because these zones are less relevant for route choice of cars, since most routes in the city centre are done by active modes (walking, cycling) and public transport. Most car trips in this network are through traffic on the major roads and to workplaces in the suburbs. Furthermore, much less information on real-time travel times from FCD (by car travellers) was available from the city centre.

The OD matrix was available for every quarter of an hour during the morning peak, between 05:30 and 11:00 hours, and for the evening peak between 14:30 and 20:00 hours. Because a continuous demand was needed for the daytime, for the quarters in between, the demand was estimated with a weighted average between each origin-destination pair. That means close to the end of the morning peak the last demand of the morning peak has a high weight and the first demand for the evening peak a low weight. The total demand over all origins and destinations for each period of the day is visualized in Figure 4.

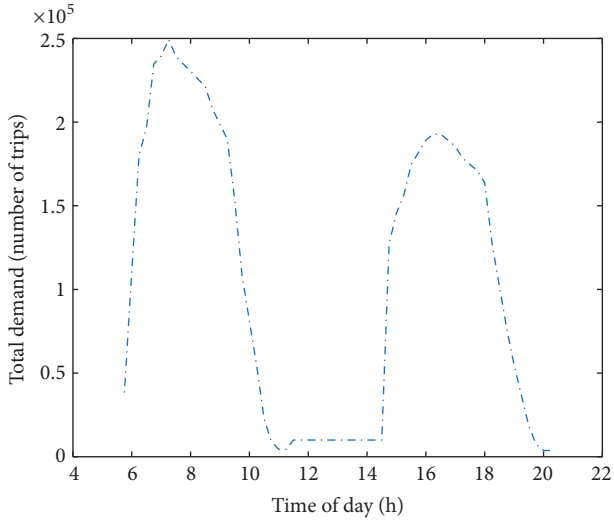


Figure 4: Total demand over the day.

5. The Smart Routing Algorithm

The smart routing algorithm in this case study consists of two parts, namely, an off-line route generation algorithm and an on-line routing advice for individual road users. In practice, the off-line route generation is done as preprocessing step, while the routing advice is determined in real-time while a user is on the road. In this case study, both are done off-line in our data laboratory.

The off-line route generation algorithm generates route alternatives between each origin-destination pair, with the purpose of being able to distribute the traffic over these alternatives and in this way improve individual and/or total travel times in the network. The route alternatives should be good alternatives for the trips of the road user. They are calculated in advance, based on historic traffic data in the network. The route generation is done based on shortest path calculations with a generalized cost function. The attributes of the link cost that are taken into account are

(i) the travel time (the lower the better),

(ii) the total distance (the lower the better),

(iii) the safety/comfort, expressed as the distance on the underlying road network (not motorways) as percentage of the total route length (the lower the better),

(iv) the comfort, expressed as the average capacity per kilometre (the higher the better: sufficient capacity, more lanes).

Each of these attributes has a weight factor to determine the overall score of the route. These weighting factors can be varied per user or user type, as was done in the PPA [8]. Here we used fixed weighting factors $w_1 = 1$, $w_2 = 0.25$, $w_3 = 0.20$, and $w_4 = 0.0025$. In order to find route alternatives, first the fastest route, the shortest route, and the route with highest capacity are determined and added to the route set. In order to create more route alternatives, a Monte Carlo approach was used where the generalized costs per link are varied stochastically. The maximum number of routes that are generated per origin-destination pair is adjustable. In this case study, a maximum of ten routes was chosen. As will be explained later, all car drivers without the in-car routing device are distributed over all of these routes, while the users with the in-car routing device will be distributed over a maximum of three routes with the shortest real-time travel time.

The on-line routing advice will provide a real-time routing advice at departure time to individual road users. For each user, a selection is made from the ten route alternatives for his origin and destination. The pregenerated route alternatives are evaluated real-time with the measured travel times. Several strategies are possible. The easiest approach is to provide only the route with the shortest travel time to the user. Downside of this strategy is that if too many users receive and follow this advice, the capacity of this route might be exceeded such that this route is beginning to suffer from congestion. Another strategy is to provide a number of route alternatives to the user, from which he/she will make his own route choice, based on his personal preferences. Both strategies have been evaluated in this case study; for the second strategy the user was given a choice between 3 routes.

6. Accuracy and Availability of FCD Data

Accuracy of single FCD measurements is dependent on both the measurement accuracy of the device (e.g., GPS or WIFI and quality of the GPS receiver), the environment (e.g., high density of high buildings will deteriorate the accuracy of GPS), and the software that is used for, for example, map-matching, speed calculation, and corrections. The average position error ranges from 2 meters on an open square to 15 meters in wide streets with buildings on both sides [9]. For speed calculation a GPS usually takes a running average of data points with some smoothing function. This means that while accelerating or decelerating the GPS will have a larger error than at constant speed. Besides location measurements, signal Doppler shift is also used to make it more accurate. Speed measurements can furthermore be extracted in other ways, for example, directly from the car; however, speed extracted from GPS is usually more accurate than measured by the car, since it is not affected by inaccuracies such as the vehicle's wheel size or drive ratios. It is dependent however on GPS satellite signal quality but these errors can be minimized with the use of moving average calculations. However, GPS is more dependent on the environment than speedometers in the car, think of tunnels, high buildings, and so forth. For GPS speed accuracy based on GPS-Doppler 10-second average speed accuracy better than 5 cm/s with the confidence level better than 99.9% has been observed [10].

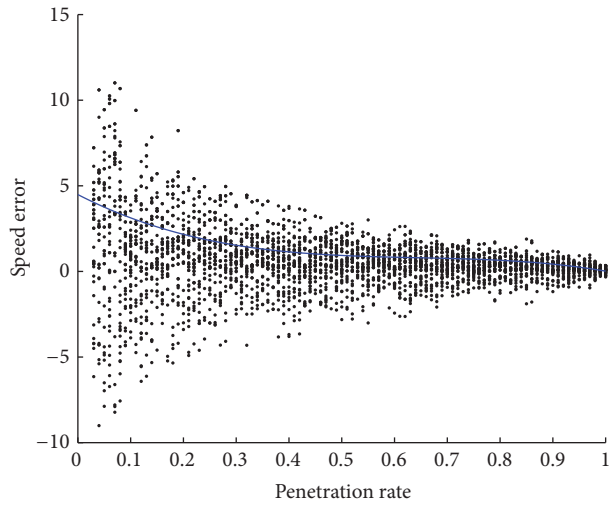


Figure 5: Estimation of percentual speed errors for the average speed at a road section with speed limit 80 km/h, moderate traffic flow, and a length of 400 m, based on FCD samples with different penetration rates. The solid curve represents the average absolute percentage error.

One could conclude that single GPS speed measurements are fairly accurate, though, for average speed calculations of a road section the accuracy of the average speed depends more on the availability of FCD than on the accuracy of single FCD speed measurements. Let us do a simple calculation exercise. For example, assume a road section of 400 meters with a speed limit of 80 km/h and a flow of 1000 veh/h, which corresponds to a density of 18 vehicles on this road section. Assuming furthermore that the real speed of these vehicles is normally distributed with mean 82 km/h and a standard deviation of 5 km/h and that the measured FCD speed has a normally distributed error with a mean of 1 km/h, the FCD speed error (difference between real average speed on the road section and the average measured FCD speed) for different FCD penetration rates is shown in Figure 5. A similar calculation is done for the network of the region of Amsterdam, separately for each link in the network, considering the speed limit, link length, and estimation of the flow, for a penetration rate of 1%, 10%, 50%, and 90%. The result is given in Table 1. For 1% the density is in most cases too low to be able to estimate the error; therefore no estimation could be made for this penetration rate.

In addition to the speed accuracy, low FCD penetration rates have an additional problem; namely, during the

measurement period at (short) road sections not any FCD vehicle might be present. The probability for this can also be calculated statistically; assuming constant speeds one can use the Poisson distribution for this. This probability is furthermore dependent on the duration of the time interval, the flow (q), the link length, and the average speed. Vehicles present on the link at the beginning of the interval are also included by extending the time interval with the average travel time. The results for a three-lane motorway section (100 m) and an urban road section are shown in Figure 6 (notice the difference in axes scale). Notice that, on the motorway with moderate and high flow, already above 2% penetration rate the probability that there is no FCD available is negligible. On the urban road, this is around 20%.

However, often one wants to have more than one observation on a link in order to calculate an average speed which is accurate enough (to average out speed dynamics on the road section and measurement inaccuracies or to remove odd cases such as vehicles resting at a parallel parking space). The same calculation can be done for the probability of at least 3 FCD vehicles to be present on a road section during a given time interval. The results are shown in Figure 7. This led to quite different results; on the motorway the probability that there are less than 3 FCD available is negligible above 4% penetration rate and on the urban road above 40% penetration rate.

Taking this exercise further to estimate the relationship between the penetration rate of FCD vehicles and the availability of reliable average link speeds in a traffic network, we used this calculation method together with the network data of our case study around Amsterdam with measured and estimated speeds and flows for all links (12,425 links, average link length 186 m), on the January 28 at 8:00, assuming a uniformly distributed FCD penetration rate throughout the network. For each link, the probability that sufficient FCD is available is estimated and summed up for the whole network. From this, the expected number and percentage of available links are calculated. The result is shown in Figure 8. Relating the link availability percentages to FCD penetration rates, a link availability of 10%, 50%, and 90% corresponds to, respectively, 1.7%, 5.8%, and 15.6% FCD penetration rate throughout the network. From this one can conclude that already quite low FCD percentages lead to relatively high availability of links where a reasonably accurate average speed can be calculated. The other way around, looking at the FCD penetration rates as used later in this paper, we get the percentages as given in Table 1.

Table 1: Relation between penetration rate of FCD, speed error, and link availability.

Penetration rate of FCD	Speed error (%)	Link availability (%)
1%	NaN	5%
10%	5.6%	76%
50%	4.6%	99%
90%	1.1%	99.5%

7. Modelling Smart Routing

In this paragraph, we explain in more detail how the route choice was modelled for both "normal" users (those who do not have the smart routing system to their disposal) and smart routing users, as well as which days were modelled and how the quality variations in the traffic data are modelled.

7.1. Modelling Normal Users. We assume that the "normal users" do not have real-time information about the actual speeds on the road network but that they have a notion of what the speeds usually are on their routes, derived while driving in the network during the recent past. In the model, this is estimated by calculating the mean speeds on each link in the network over the last two months. The route choice of these users is based on the average travel times on their routes from these average speeds (without the speed of the current day). The route choice is modelled with a multinomial logit model. The multinomial logit model (MNL) is the most widely used choice model, due to its simple mathematical structure and ease of estimation [11]. In this model, the probability for using route r is calculated as follows:

$$P(r) = \frac{e^{-\theta T_r}}{\sum_r e^{-\theta T_r}} \quad (1)$$

in which T_r are the generalized costs (in this case the estimated travel time) of the route and θ is a scaling parameter which represents the knowledge level of the user. In this case study, we used the following settings: theta = 1, T_r travel time in minutes, which has been proven in earlier studies to be realistic settings [12]. More sophisticated route choice models exist which, for example, correct for overlapping of routes (see [11, 13, 14]). This case study focuses on the effect of inaccurate input data rather than on route choice modelling. A route choice model that takes overlapping of routes into account could also represent the effects of inaccurate information on route choice better; however, in the current study this is not (yet) done, since this would lead to much longer calculation times, while the calculation time is already a limiting factor in this study due to the large number of variants in information levels. Also, the route generation model already filters out routes that greatly overlap. Therefore it was accepted that the multinomial route choice model is sufficient for the purpose of this study.

The route flows for every OD pair and every time period are then calculated by multiplying the probabilities with the number of trips per OD pair.

7.2. Modelling Smart Routing Users. Between each OD pair, we derived at most ten alternative routes, as mentioned in Section 5. A certain fraction of all drivers uses the smart routing application, which we call the penetration rate p of the system. The follow-up rate of the system is already contained in this fraction, but can be modelled separately when desired. Users with the smart routing application will get a routing advice representing one or more routes to their destination. Here we investigated first the scenario that only

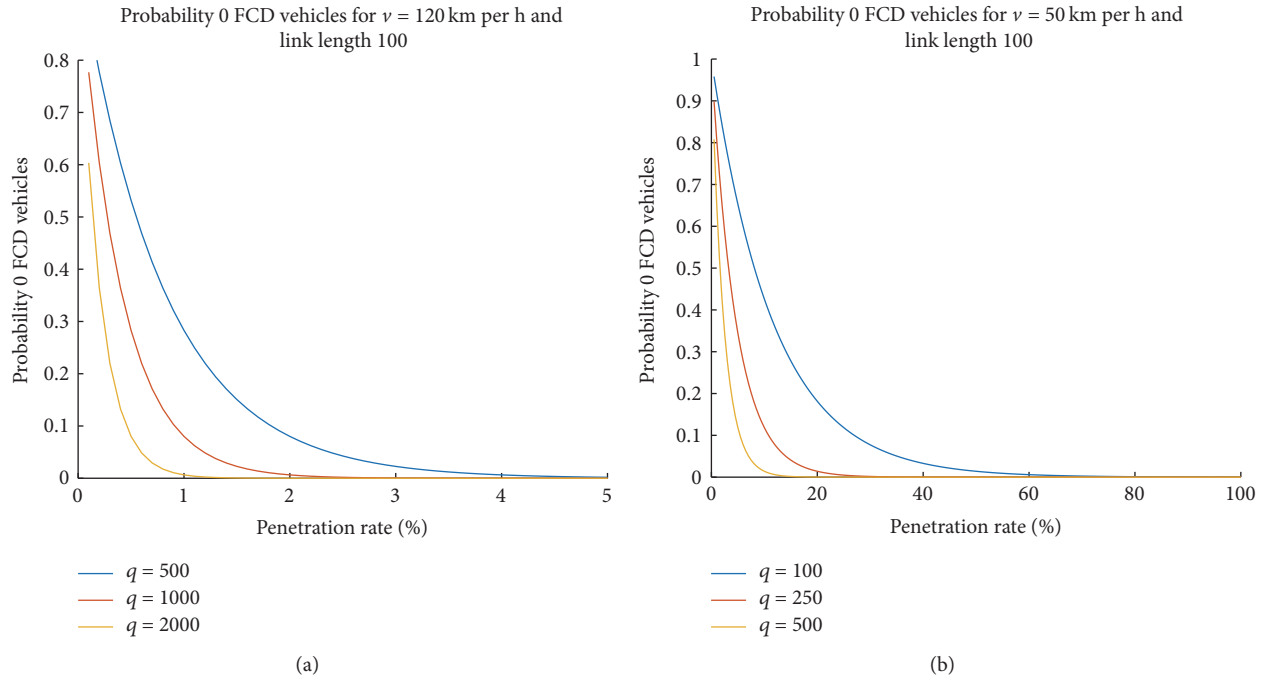


Figure 6: Estimation of the probability that not any floating car is observed during a 5-minute time interval on a road section of 100 m length and different lane flows (q), on a 3-lane motorway with speed limit 120 km/h (a), and on a one-lane urban road with speed limit 50 km/h (b).

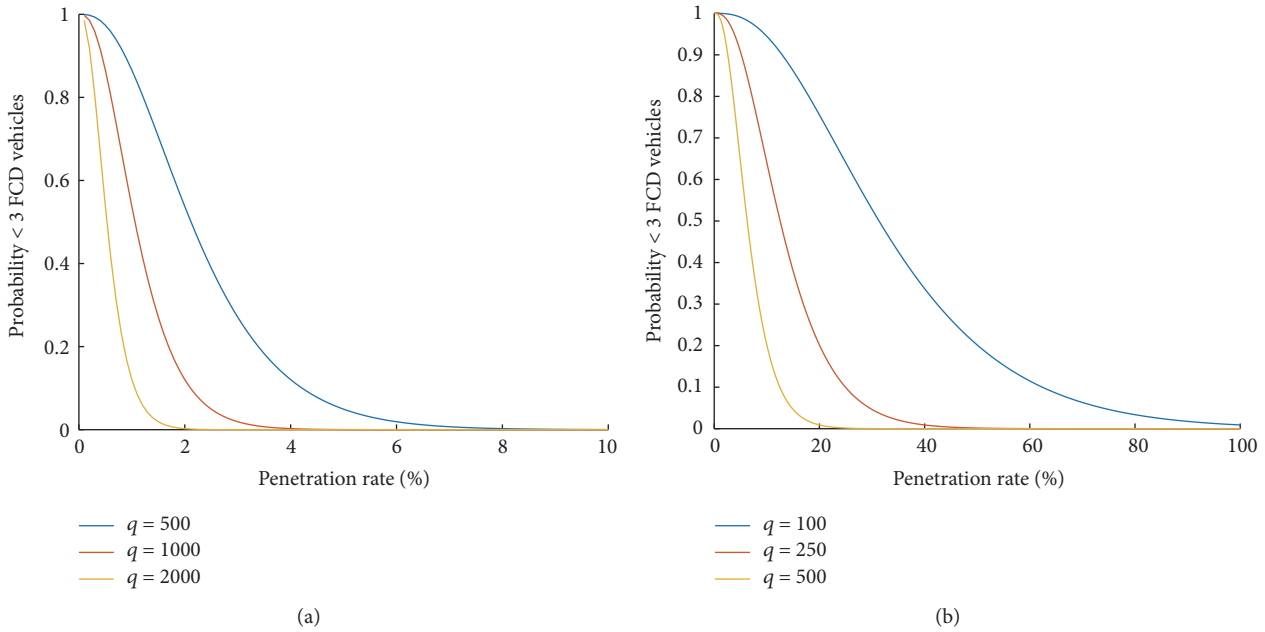


Figure 7: Estimation of the probability that less than 3 FCD vehicles are observed during a 5-minute time interval on a road section of 100 m length and different lane flows (q), on a 3-lane motorway with speed limit 120 km/h (a), and on a one-lane urban road with speed limit 50 km/h (b).

the route with the shortest actual travel time is advised to the users, which they will accept. Secondly, we investigate the scenario that the three best/fastest routes based on the actual travel time are shown to the user. The users route choice is assumed to be distributed over this top 3 conform the multinomial logit model, similar to the users without smart routing advice, but in this case based on the actual speeds instead of average speeds. The routing advice does not take into account the capacity and actual flows on the routes. This is to conform most current routing advice systems work, since it is a difficult task to estimate actual flows in the network and to determine a routing advice based on, for example, a user

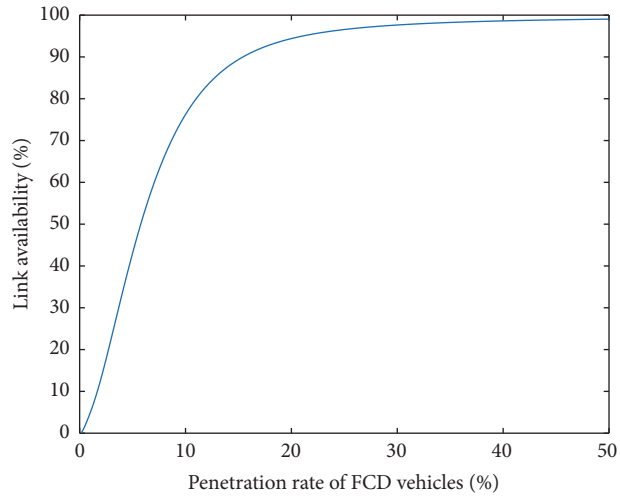


Figure 8: Estimation of the link availability for different penetration rates of FCD vehicles on the study network of the region of Amsterdam.

equilibrium in order to prevent that too many drivers choose the same route leading to congestion. But we do take into account the fact that the resulting flows might lead to longer travel times when certain route parts approach or exceed their capacity, using the so-called BPR function [15]:

$$\mathrm{tt}_{\mathrm{cong}} = \mathrm{tt}_{ff} \cdot \left(1 + \alpha \left(\frac{q}{C}\right)^{\beta}\right). \quad (2)$$

The BPR function is calibrated on the speed data in the network on link level (for each link separately) with a polynomial fit, for the links for which measured speeds were available. Capacity of the links was estimated based on the speed limit and the number of lanes. For the other links, default values for alpha and beta were used: alpha = 0.15 and beta = 4.

7.3. Selection of Days. The smart routing advice is expected to have a larger impact when the traffic situation is different from the daily traffic pattern, that is, when there is more congestion than normal, because then it has more added value to know the actual traffic situation. Therefore, we want to test the system for one or more days where the traffic speeds have a large deviation from the speeds on an average day, and for comparison also for a day which has speeds close to the ones on an average day. In order to find such days, the average network speed was calculated for each period of the day and all days in the first three months of 2015 for which sufficient data were available. Next, the average network speed was calculated over all weekdays (Monday to Friday). Days with a lot of outliers were excluded from the calculation of the average network speed. For each day, the network speed was compared with the average speed and a choice was made for an average day and for two (different) abnormal days. One of the chosen abnormal days is Tuesday, February 3, 2015. On that day there was a lot more congestion than usual. This was caused by slipperiness due to the snowy winter weather during the whole day. The network speed of the "abnormal" day together with the network speed of the average weekday is shown in Figure 9(b), where it is shown that the actual network speed is much lower than the average network speed. The other abnormal day was the 5 February 2015. On that day there was also a lot more congestion than usual, caused by several incidents, especially during the morning peak, as shown in Figure 9(c). The selected average day is Wednesday, January 28, as is shown in Figure 9(a). A remark needs to be made that in exceptional days like those investigated, demand and capacities are different than normal days, and also uninformed users' behaviour may be different, since they will face different delays. However, we assume that most users will still take the route that they are used to take and therefore the estimated route choices are considered representative also for these cases.

7.4. Variation in Quality of the FCD. Floating car data is never 100% correct. In this study we want to know what the effect is of different levels of quality of the floating car data. There are different types of errors or inaccuracies in floating car data, but in this study we focus on two types of inaccuracies: lack of data and inaccurate data.

For the first inaccuracy type (lack of data) the number of links for which the actual traffic speed is known is varied. To be able to generate a complete advice, for the links for which no information is available, the free flow speed of the link is used. The percentage of links for which no information is available is varied from 0% to 100% with steps from 10%. The links for which no actual traffic speeds are available are drawn randomly with a higher weight for links with a higher flow. The flows are used as weights in a drawing without replacement. This is done in this way because in practice there is also a higher probability that floating car data are available on links where more vehicles have passed. The smart routing application will base its routing advice on the adapted, incomplete information of the speeds in the network. Since part of the congestion is not observed, these travel times will generally be shorter than the real travel times. However, it may still be better than the average travel times that are used for the route choices of the normal road users without the smart routing application.

For the second type of inaccurate data, we assume that the inaccuracy of the speeds found in the floating car data is normally distributed with mean value of the real average speed and a standard deviation that is varied from 0 to 0.9 with steps of 0.1, as shown in Figure 10. For each link, a separate drawing is done and applied to the speed of that link. The smart routing application will again base its routing advice on the adapted (inaccurate) information of the speeds in the network. The resulting travel times may be either larger or shorter than the real travel times, but again may still be better than the average travel times that are used for the route choices of the normal road users without the smart routing application.

8. Results

8.1. Results Using Perfect Information. Suppose we have access to perfect traffic information and we offer the smart routing

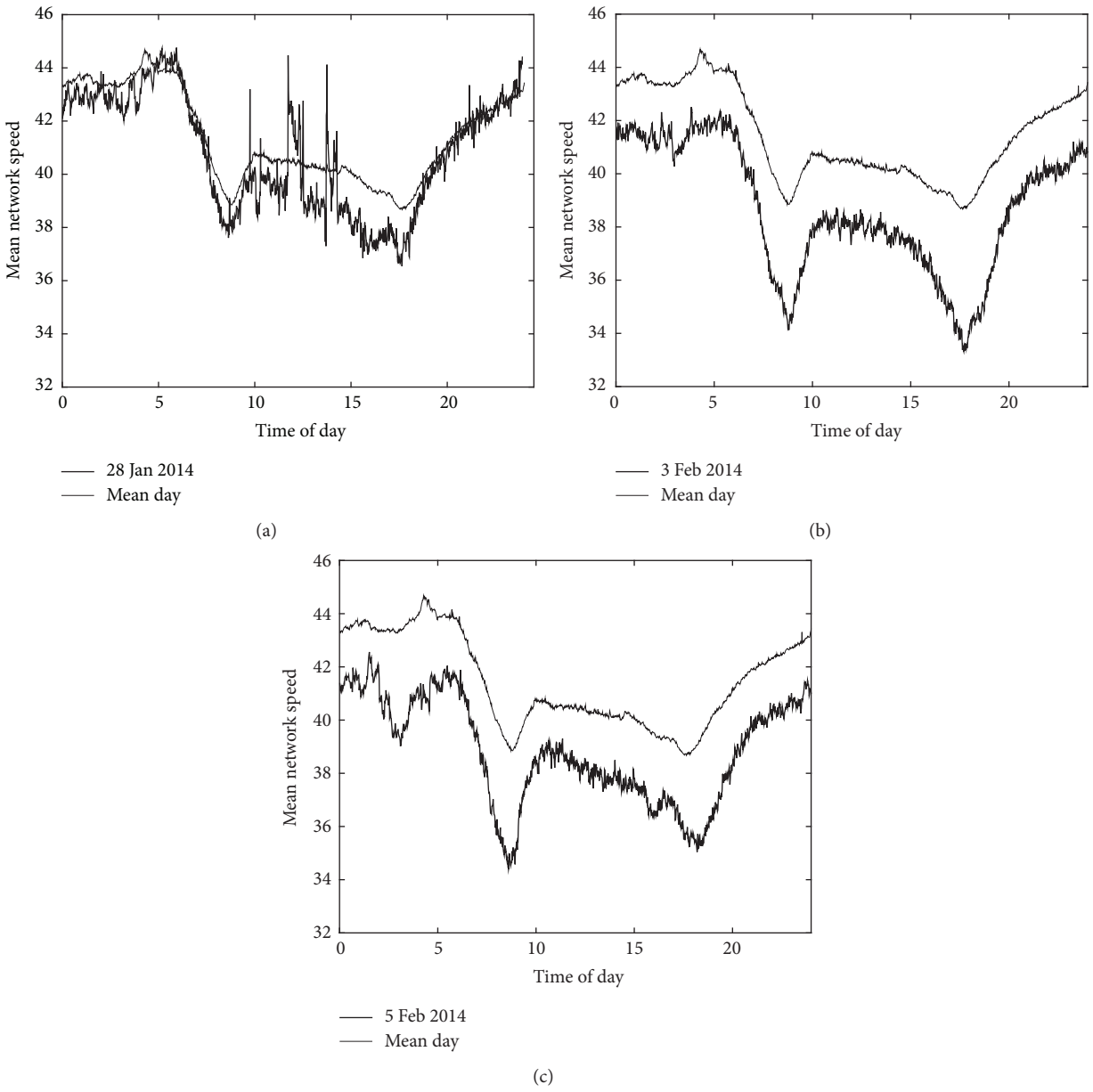


Figure 9: Selected day with an average speed pattern (a), a day with an abnormal speed pattern February 3, 2015 (b), and a day with an abnormal speed pattern on February 5, 2015 (c).

advice as explained above. Then we can calculate what the potential effect is of the smart routing application. Since the system knows the actual travel times, the total travel time in the whole network is expected to decrease, as well as the delays in the network. On individual level, smart routing users will benefit as well. Only when too many road users choose the same route, based on the advice, may (individual) travel times increase.

We have calculated the effects for different penetration rates of the system for the three different days. The total delay was calculated as the difference between the total of all free flow travel times in the network and the total of all actual travel times in the network. The difference in total delay without smart routing and with smart routing is compared for several penetration rates of the system (1%, 10%, 50%, and 90%). The result is shown in Figure 11. This figure clearly shows that the improvement of the system is largest for high penetration rates of the system and during the peak hours, as can be expected.

In Figure 12, the total delay improvement over the whole day is shown for the three different days and the two route advice strategies (showing only the best route or the top 3 best routes). In Figure 12(a) it can be clearly seen that the day with the most deviating congestion (February 5) has the largest benefit in saving delay hours, as expected. It is striking that the results show a clear linear relationship with the penetration rate. It is difficult to understand this relationship; it seems that, despite the complexity of the network and the

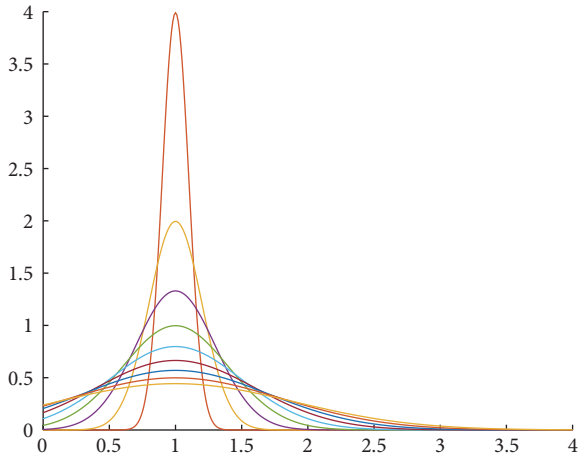


Figure 10: Error factors for the inaccuracy of the measured speeds.

nonlinearities in the overall system, the impact of more FCD in the system can be approached with this linear relationship, though it could also be the case that the way of modelling and the assumptions used oversimplify the outcomes. We however still believe that this will not change the general conclusions, order of magnitude of the effects, and other general insights gained from this study.

For a penetration rate of 10% on the average day with the best routing advice, the total improvement over the whole day is 4480 hours, or 0.8% of the total delay in the network as shown in Figure 12(b). Giving a top 3-route advice gives a little bit less good results of 3150 saved hours or 0.8%. Apparently the possible prevention of congestion on the best route does not compensate the longer travel times of the second and third best routes. For 90% penetration rate, 4.7% to 7.0% of the total delay in the network would be saved on the average day. It is also striking that the results of the average day and the day with the most congestion do not differ much percentagewise for the best routing advice only (8.6%).

Using a value of time of EUR 12.28 (assuming an average of 10% freight traffic and the key figures from [16]), this means a saving of 55,000 euros for the average day. This can be scaled up to a yearly level by calculating with the yearly number of weekdays (261 in 2015). This gives an estimate of 14.5 million euros saved on a yearly basis for 10% penetration rate of the system, when perfect traffic data and traffic information would be available. For 90% penetration rate 130 million euros could be saved on a yearly basis, calculating only with average days. However, since part of the days has more congestion than average, the potential is larger.

8.2. Results with Variation in Quality of the FCD. In Figure 13, the relative improvement in delay time compared with no routing advice is shown for the average day (January 28, 2015). (a) and (c) are the results where the system only gives the best route option to all users. In (b) and (d), the results when the 3 best routes are given are shown. In (a) and (b), the accuracy of the speed information is varied, while the graphs in (c) and (d) show the results for the variation of available link speeds.

As can be seen from these results, it seems that inaccurate speed information has a higher impact on the results than incomplete information, though these are different quantities and cannot be compared exactly. Furthermore, giving the user only the best route option leads to better results than giving the user a choice between the three best route options. The same as in the case with perfect information, it appears that the possible prevention of congestion on the best route does not compensate the longer travel times of the second and third best routes. However, if the errors on the link speeds are higher than 40%, the total impact becomes negative.

For the best route option only, the highest impact is around 7% with 90% penetration rate of the system. For the three route options, the impact stays below 5%. With 10% penetration rate, the impact is below 1% and around 0.5% with three route options. For 1% penetration rate, the effects on network level are negligible.

The completeness of the information has less (negative) influence, the impact stays within 4%–7% for 90% penetration rate of the system and 1 route option, even when no actual link speeds are available. Apparently the route advices based on estimated link speeds (free flow speeds) are still better than the route choice based on the average knowledge of drivers of speeds in the network.

Figure 14 gives comparable results for February 3, 2015, where more congestion was present in the network. Surprisingly, the impacts are almost comparable as for the average day. The cause of this can be that this day the congestion was due to snow and ice during the whole day and on all roads. Therefore not much can be gained with route advices for alternative routes. Only the link availability is more important on this day, for low link penetration rates.

Finally, the results for February 5, the most "abnormal" day, is shown in Figure 15. It is clear that the effects for this day are the highest; for the speed accuracy this is 8.6% for 90% penetration rate and 1% for 10% penetration rate and when you calculate the effect during the morning peak hour the effect is about 14% for 90% penetration rate and about 1.5% for 10% penetration rate.

Assuming that the drivers that use the smart routing app also provide the FCD data, the penetration rate of the users is the same as the penetration rate of the FCD, such that the percentages of Table 1 apply. These can be linked to the results of this paragraph. For the link speed error, a translation is needed from average absolute error to the standard deviation of the error distribution. For the percentages of Table 1 (5.6%, 4.6%, and 1.1%), this is, respectively, 0.07, 0.06, and 0.01. Linking this to the results of this paragraph, the FCD penetration rate hardly makes any difference for the results based on speed error or for limited link availability, as shown in Table 2. One should however take into account the fact that the link speed error is caused by more than a small random error for GPS inaccuracy, the data may also contain outliers in the speed by other causes, or in a densely built up area the error will be larger.

Link availability and speed inaccuracy are linked in the sense that a certain minimum availability of FCD is needed on a link in order to be able to actually calculate an average speed. The higher the availability, the higher the speed

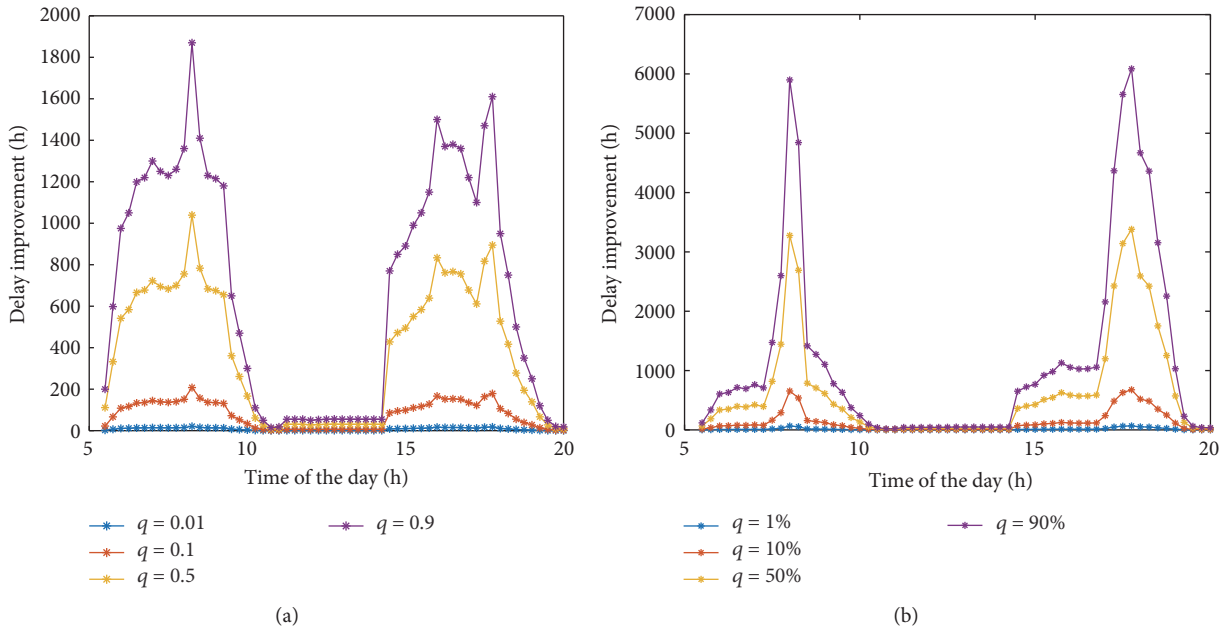


Figure 11: Effect of smart routing using perfect information for different penetration rates on the average day (a) and for February 5, 2015.

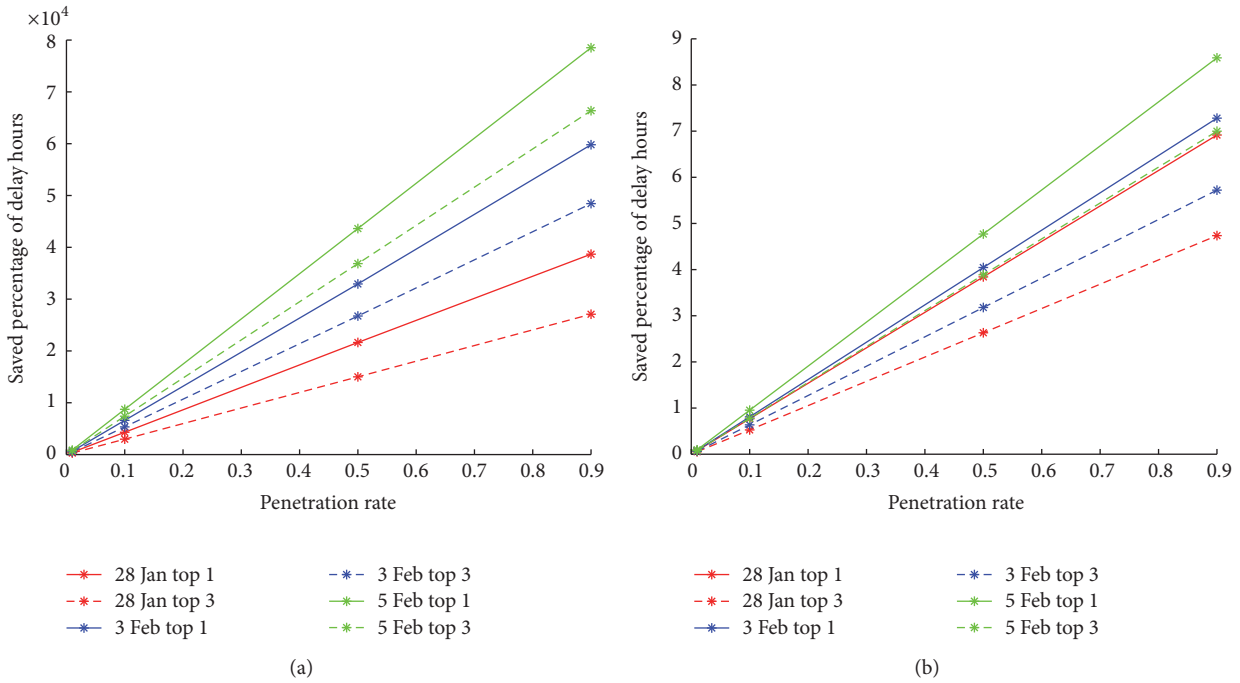


Figure 12: Saved delay hours as total over the whole day (a) and as a percentage with regard to the situation without routing advice (b).

Table 2: Relation between penetration rate of FCD, speed error, and link availability.

Penetration rate of FCD	Delay improvement for speed inaccuracies			Delay improvement for limited link availability		
	28 January	3 February	5 February	28 January	3 February	5 February
1%	Unknown	Unknown	Unknown	<0.03%	<0.02%	<0.02%
10%	0.8%	0.8%	1.0%	0.7%	0.7%	0.9%
50%	3.8%	4.0%	4.8%	3.8%	4.0%	4.8%
90%	6.9%	7.3%	8.6%	6.9%	7.3%	8.6%

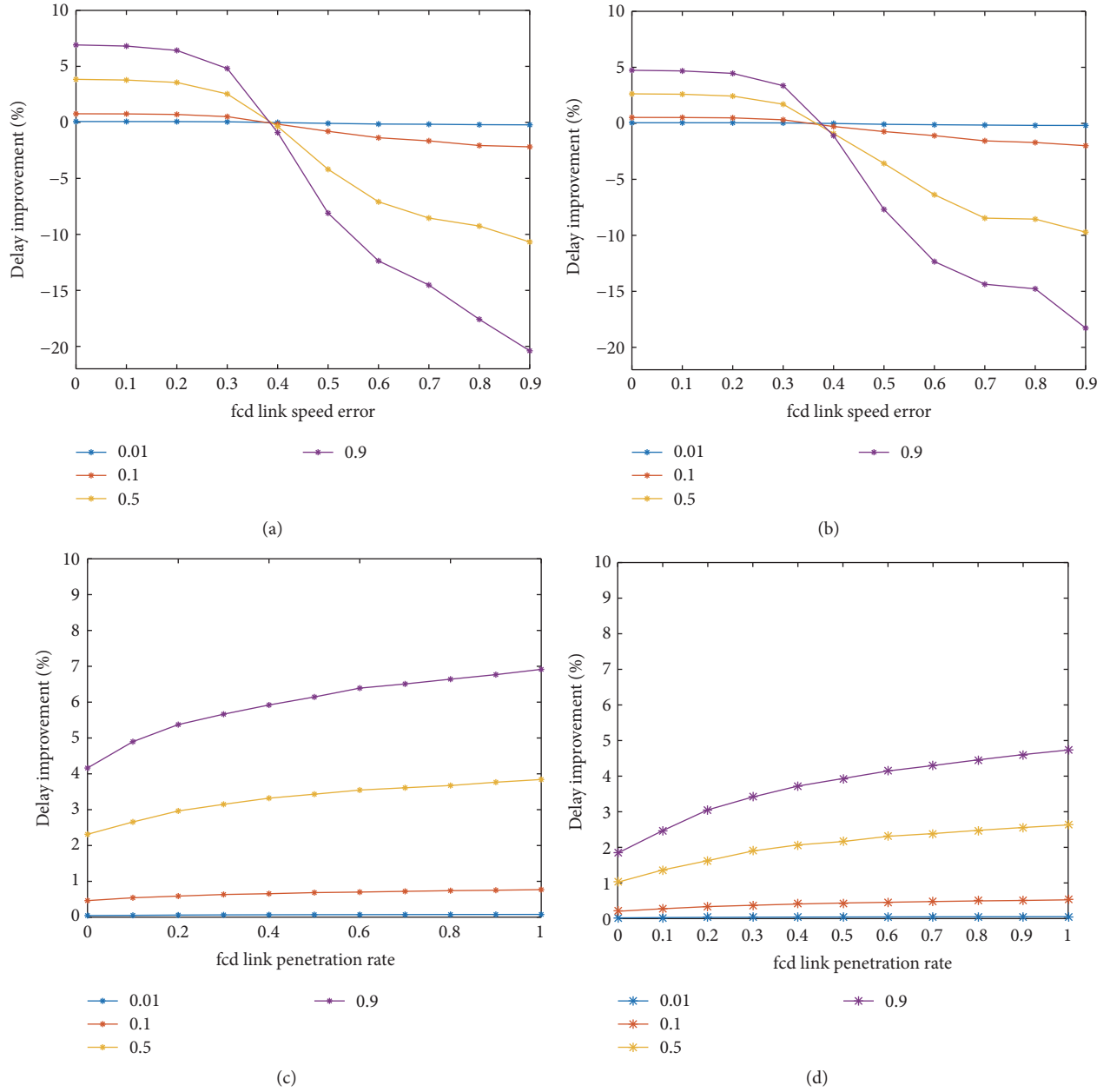


Figure 13: Delay improvement for 28 January 2016 (average day) as percentage with regard to the normal delay: variation in speed accuracy with best route advice (a), variation in speed accuracy with top 3-route advice (b), variation in link speed availability with best route advice (c), and variation in link speed availability with top 3-route advice (d).

accuracy. For 50% and 90% FCD, the results are almost equal, because they are both very close to the maximum effect. An FCD percentage somewhere between 1% and 10% would show larger differences. Actually one should take into account both quality aspects in order to estimate the total effect; the combined effect is smaller than when taking into account only one of the inaccuracies.

9. Relation with Data Quality and Cost Benefit

Based on the results this case study allows for a cost-benefit analysis which relates the amount and quality of the input (floating car) data to the improvement of the individual and network performance. This may be used to support investment decisions on such a system to answer questions such as what can be the benefit of real-time in-car routing, which implementation rate should be achieved in order to reach a certain improvement, and which quality of input data is needed (either from the participating vehicles or from another source)?

The benefits of the system are calculated for different penetration rates and qualities using a value of time in the same manner as above for the case with perfect information. This gives pictures of similar shape of Figures 13–15, up to

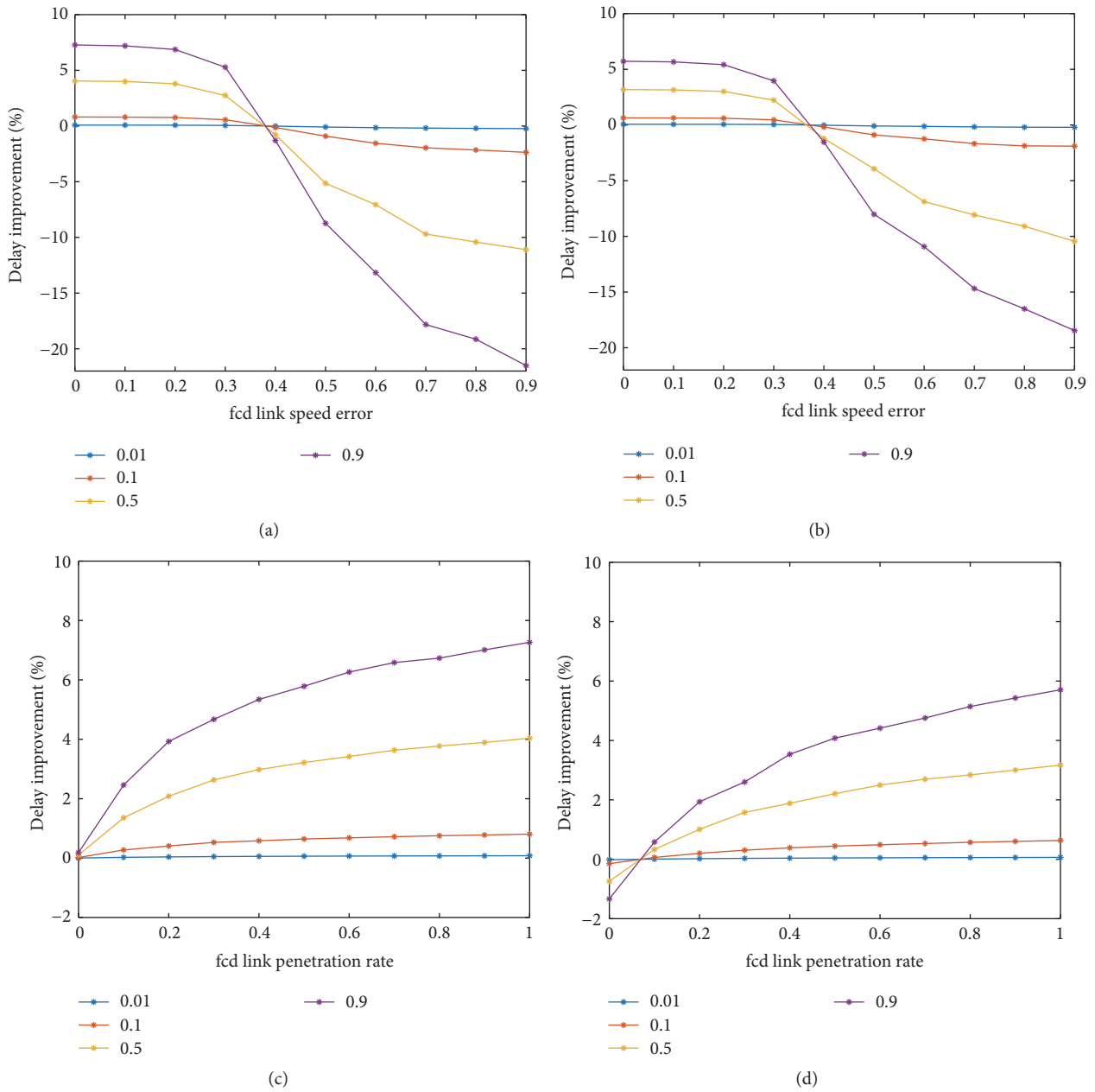


FIGURE 14: Delay improvement for 3 February 2016 (day with a lot of congestion) as percentage with regard to the normal delay: variation in link penetration with best route advice (a), variation in link penetration with top 3-route advice (b), variation in speed accuracy with best route advice (c), and variation in speed accuracy with top 3-route advice (d).

one million euros a day for 90% penetration rate on February 5.

The benefits can now be weighed against the costs of such a system and data gathering. These costs consist mainly of the software development, maintenance, and data communication costs. The data communication costs depend on the penetration rate of the system and on the sample rate: the higher the penetration rate and the higher the sample rate are, the more the communication is needed. However, in the case of a mobile phone service, nowadays most people have a fixed cost subscription with a large amount of data communication included. Though we can assume that some quality aspects of the data are related to certain costs.

The total number of trips during one day in or crossing the region around Amsterdam in this case study is 6.4 million, based on the OD matrix (night not included, see Figure 4), or 2 million during the morning peak. Since most people make more than one trip a day, namely, 2.7 on average [16], the total number of different people making one or more trips in this area can be estimated at 2.4 million on a daily basis. Hence a penetration rate of, for example, 10% relates to approximately 240 thousand people (selected from the people

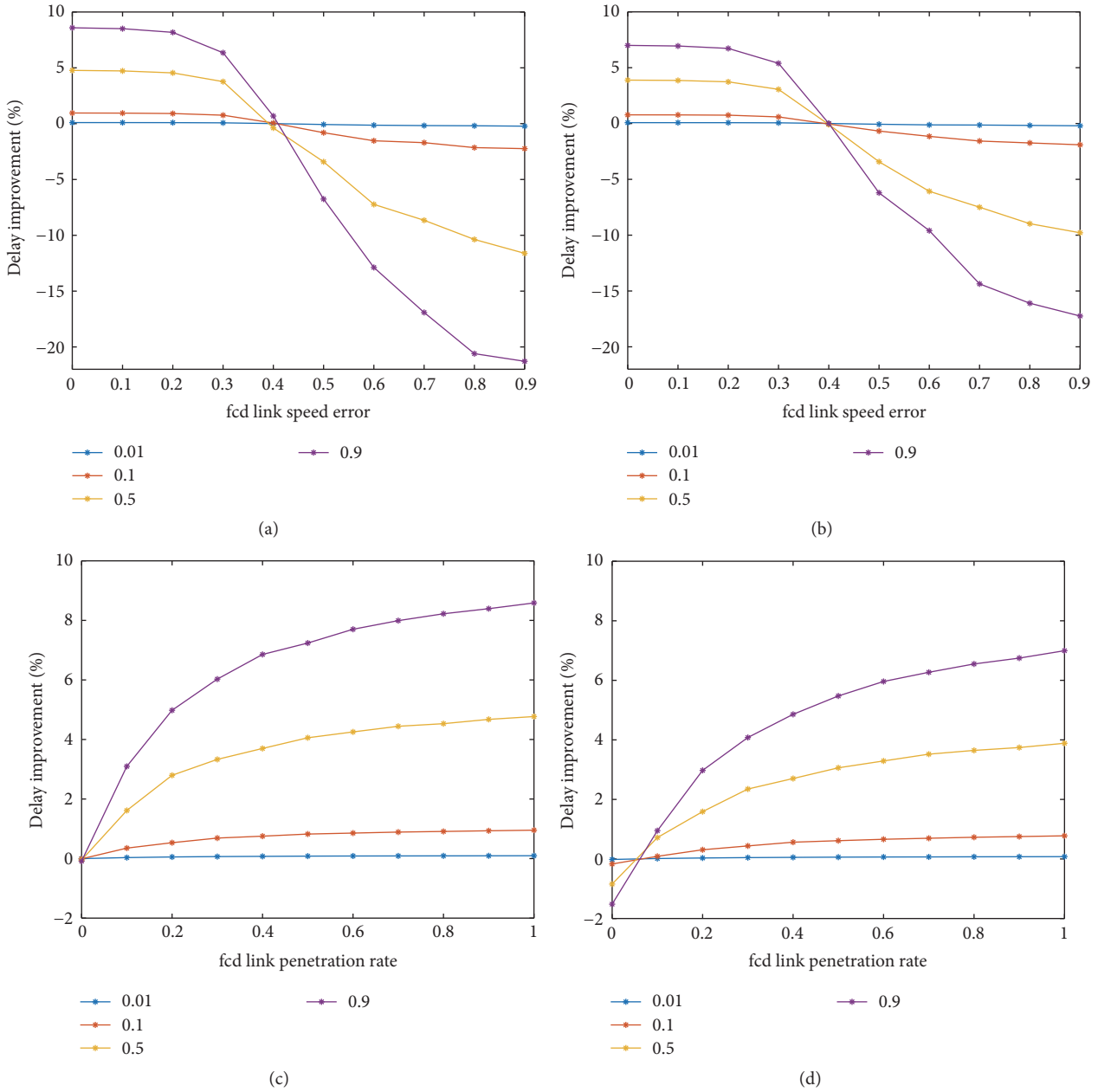


FIGURE 15: Delay improvement for 5 February 2016 (day with a lot of congestion) as percentage with regard to the normal delay: variation in link penetration with best route advice (a), variation in link penetration with top 3-route advice (b), variation in speed accuracy with best route advice (c), and variation in speed accuracy with top 3-route advice (d).

that are known to travel in or through this area). For the morning peak this is comparable, assuming that one usually makes only one trip during the morning peak, one gets 200 thousand people for 10% penetration rate. Assuming that they all need a mobile phone subscription with data for an average cost of 20 euros per month, this gives a total cost of around 200 thousand euros per working day. This does not weigh against the benefit in value of time per day that was calculated earlier (55000 for a normal working day); however, since most people already have a mobile phone subscription with data connection, the extra costs for this service are zero.

Additional costs can be considered for the data collection, analysis, and processing, which consist of a fixed part and a part dependent on the amount of data. Twice as much data usually needs more processing and analysis time, but not twice as much time, since certain analysis steps are as much work for a small dataset as for a large dataset, and independent calculation steps can be parallelized. The same scripts can often be used to handle a larger dataset. So assume, for example, that every month a fixed amount and a variable amount of man hours are used to handle the collected data correctly, say 10 hours (fixed) plus one hour for

the data of every 10 thousand users for a tariff of 100 euros per hour, giving labor costs of 3000 euros per month. Then some costs for hardware and software could be somewhere around 1000 euros per month (considering, e.g., write-off of advanced computing servers or subscription for cloud computing and storage). Now the benefit falls greatly to the advantage of the smart routing application. However, initial costs for development of this service (to make it mature and ready for large-scale use) are also substantial and can, for example, be estimated around 100 thousand euros. A business model could be used to ask a small amount from the users for this service (let us say 2 euros for downloading the app and 2 euros per month for using it), in order to gain back the initial development costs quickly.

10. Conclusions

The results of this study give insight into the size of improvement that is possible with individual in-car routing advice, for different traffic situations, and which quality of the input traffic data is needed to achieve a certain degree of improvement. This improvement is expressed with traffic performance indicators as total travel time and total delay.

For a penetration rate of 10% on the average day, when perfect traffic data and traffic information would be available, the total improvement over the whole day is 0.8% of the total delay in the network, which means a saving of about 15 million euros on a yearly basis. For 90% penetration rate, 7.0% of the total delay in the network would be saved, which is approximately 130 million euros on a yearly basis.

Though the previous results are based on perfect information, this study furthermore shows that even with low data availability and low speed accuracies the delays in the network are improved. For 90% penetration rate, on a regular day the improvement in saving delay hours is about 7%; on a day with incidents this increases to 8.6%. When only looking to the morning peak, the difference in effect for a normal day or a day with incidents is larger: from 4% on a regular day to 14% on a day with incidents.

When the link speed accuracy is less than 40%, the effect of the smart routing advice becomes negative, because it is based on the wrong information.

Also a relation has been made between the FCD penetration rate and the quality aspects of the data for the smart routing: with a penetration rate of at least 10%, the speed error based on small individual location errors from GPS stays below 6%. Therefore the FCD penetration rate has hardly any influence on the delay improvement from the smart routing. For a penetration rate of 1% this could not be estimated because of very little data. For the link availability, an FCD penetration rate of 1% leads to link availability of 5%. For a penetration rate of 10% or higher, the link availability is so high that again the FCD penetration rate has not much influence.

A coarse cost-benefit estimation has been done based on rough estimates of various costs (fixed and variable). This shows that the necessary communication costs for a mobile data connection (subscription with data for the users) do not weigh against the benefit in value of time per day, however, since most people already have a mobile phone subscription with data connection, the extra costs for this service are zero. Maintenance costs are very low compared to the benefits. Initial development costs are estimated substantial but can be gained back quickly by asking a very small amount from the users for this service.

11. Recommendations for Further Research

Though in this case study a good attempt has been made for the estimation of the potential effects of a large-scale smart routing service, some issues could be studied in more detail in order to improve the reliability and applicability of the results. Additional questions and steps for further research are as follows:

(i) In order to get more reliable results on a yearly level, scale up the results by using more days with different traffic patterns and investigate how many abnormal days with more congestion than average usually occur throughout a year.

(ii) Use a route choice model that takes into account overlap of routes, calibrate with real data.

(iii) Calculate personal benefits (in addition to network effects)

(iv) Improve the routing advice taking into account capacity constraints: when the route with the current shortest travel time almost exceeds capacity, advise the 2nd (or 3rd) best route. This can, for example, be done by performing an equilibrium traffic assignment based on actual traffic measurements.

(v) Improve the routing advice by updating the advice while driving, using actual traffic measurements. This was already implemented in practice in the Practical Trial Amsterdam [2].

Since this study shows that the penetration rate and FCD data accuracy are key elements in relation to the delay improvement, strategies that could increase the penetration rate of such in-car services and possibilities to improve data quality from FCD are relevant topics for further research as well. For example, additional research can be done on the accuracy of the localization: does localization based on gsm signal strength provide sufficient accuracy or is GPS accuracy needed? In the first case penetration rates are expected to be higher since not everybody has or uses GPS localization all the time on his mobile phone.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is jointly funded by TNO, Netherlands Organisation for Applied Scientific Research, and TrafficQuest, a

joint collaboration between TNO, Delft University of Technology, and Rijkswaterstaat, highway agency of the Dutch Ministry of Infrastructure and the Environment.

References

[1] T. Djukic, I. Wilmink, E. Jonkers, M. Snelder, and B. van Arem, "Exploratory analysis of traveller's compliance with smartphone personal route advice: a field trial Amsterdam," in *Proceedings of the 95th Annual Meeting of the Transportation Research Board*, 2016.

[2] I. Wilmink, E. Jonkers, M. Snelder, and G. Klunder, "Evaluation results of the Amsterdam, Netherlands, practical trial with in-car travel and route advice," *Transportation Research Record*, vol. 2621, pp. 38–45, 2017.

[3] G. A. Klunder, H. Taale, L. Kester, and S. Hoogendoorn, "The effect of inaccurate traffic data for ramp metering: Comparing loop detectors and cameras using information utility," in *Proceedings of the 19th World Congress of the International Federation of Automatic Control*, pp. 5075–5082, Cape Town, South Africa, 2014.

[4] F. Viti, S. P. Hoogendoorn, L. H. Immers, C. M. J. Tampère, and S. H. Lanser, "National data warehouse how the Netherlands is creating a reliable, widespread, accessible data bank for traffic information, monitoring, and road network control," *Transportation Research Record*, no. 2049, pp. 176–185, 2008.

[5] V. Sethi and F. S. Koppelman, "Incorporating complex substitution patterns with distance and variance scaling in long distance travel choice models," in *Travel Behavior Research, the Leading Edge*, D. A. Hensher and J. King, Eds., Pergamon Press, Oxford, UK, 2001.

[6] S. C. Calvert, M. Snelder, T. Bakri, B. Heijligers, and V. L. Knoop, "Real-time travel time prediction framework for departure time and route advice," *Transportation Research Record*, vol. 2490, pp. 56–64, 2015.

[7] H. Taale and S. P. Hoogendoorn, "Network-Wide Traffic Management with Integrated Anticipatory Control," in *Proceedings of the 91st Annual Meeting of the Transportation Research Board*, Washington D.C., USA, 2012.

[8] I. Wilmink, E. Jonkers, A. Jöbsis et al., "Eindrapport Evaluatie Praktijkproef Amsterdam," in *CAR–Perceel Regulier Verkeer*, TNO 2016 R10044, TNO, Delft, Netherlands, 2016.

[9] M. Modsching, R. Kramer, and K. ten Hagen, "Field trial on GPS Accuracy in a medium size city: The influence of built-up," in *Proceedings of the 3rd workshop on positioning, navigation and communication (WPNC'06)*, 2006.

[10] T. Chalko, "Estimating accuracy of GPS doppler speed measurement using speed dilution of precision (SDOP) parameter," *NU Journal of Discovery*, vol. 7, pp. 4–9, 2011.

[11] C.-H. Wen and F. S. Koppelman, "The generalized nested logit model," *Transportation Research Part B: Methodological*, vol. 35, no. 7, pp. 627–641, 2001.

[12] H. Taale, *Integrated anticipatory control of road networks: A game-theoretical approach [Ph.D. thesis]*, Delft University of Technology, 2008.

[13] E. Cascetta, A. Nuzzolar, F. Russo, and A. Vietta, "A modified logit route choice model overcoming path overlapping problems," in *Proceedings of the the 13th International Symposium on Transportion and Traffic Theory*, pp. 697–711, Lyon, France, 1996.

[14] S. Hoogendoorn-Lanser, R. van Nes, and P. H. L. Bovy, "Path size and overlap in multimodal transport networks," in *Proceedings of the 16th International Symposium on Transportation and Traffic Theory*, pp. 63–84, College Park, Md, USA, 2005.

[15] S. C. Dafermos and F. T. Sparrow, "The traffic assignment problem for a general network," *Journal of Research of the National Bureau of Standards*, vol. 73B, pp. 91–118, 1969.

[16] P. Warffemius, "De maatschappelijke waarde van kortere en betrouwbaardere reistijden. Kennisinstituut voor Mobiliteitsbeleid, 2013".