Releasing a Traffic Light Assistance Application for Public Testing

Michael Krause[1(✉)], Walid Fourati[2], and Klaus Bengler[1]

[1] Institute of Ergonomics, Technische Universität München, Garching, Germany
{krause,bengler}@tum.de
[2] TRANSVER GmbH, Munich, Germany
fourati@transver.de

Abstract. The paper's main focus is the requirements and implementations for creating a downloadable traffic light application from a preexisting interaction prototype. The background is the preparation of a public experiment on a test section of a rural road. The paper also contains a proposed taxonomy for classifying Traffic Light Assistants (TLAs). According to this taxonomy, the planned experiment is categorized as a technical feasibility study combined with public participation (field test) in real traffic on a rural road, with a bidirectional internet-connected smartphone application, which implements a full-featured, single-way TLA for one/next traffic light. The signal plans are determined, but should be dynamically switched and optimized depending on traffic. The smartphone-based floating car data doesn't influence the traffic lights, but should improve the speed recommendations for other cars with regard to waiting queues. The TLA is partly context aware (waiting queues, fixed speed limits) and features a speed alert.

Keywords: Traffic lights · Green light · Assistant · Application · App · Field test

1 Introduction

The work presented is part of the European Project (FP7) Local4Global. Local4Global is not limited to traffic lights assistants (TLAs). Section 3 explains how the TLA is embedded into the system-of-systems optimization project.

Many studies examine TLAs. [1], for instance, offers an overview of some of them. Section 2 provides a possible taxonomy for categorizing TLAs. The graphical user interface for the TLA used here was thoroughly assessed during a two-year-long pilot project. Section 4 sheds light on this preparatory work. This paper focuses mainly on the thoughts, requirements, and chosen implementations involved in converting the interaction prototype into a public, downloadable application (Sect. 5).

2 Taxonomy of Traffic Light Assistants

Studies and experiments involving TLAs can be classified according to modes and contents. One category often seen in the field of traffic engineering is *computer* *simulations*. The researcher makes assumptions about an equipped vehicle's behavior and programs these into a traffic simulation. A parameter often manipulated is the penetration rate. Similar experiments were also part of the Local4Global traffic use case [2]. *Technical feasibility* studies (e.g., proof of concept) constitute another field. Here, different ways can be distinguished for transmitting traffic-light information to the vehicle, *Dedicated Short Range Communications* (DSRC), an *Internet connection* (e.g., to a traffic center), or hybrid solutions. Another aspect is the ecosystem: The TLA could be an original *on-board system* or an *application*, e.g., on an after-market satnav or smartphone. That a hybrid (downloadable application running and visualized in on-board systems) is also possible in the near future is foreseeable.

The traffic lights associated with the TLA can also play a role in categorization. Signal plans can be (simplified) *fixed* or *traffic actuated*. More detailed traffic-engineering classification can become far more complicated (rule based, coordinated, public transport/emergency prioritization, etc.). The information database can be *deterministic* (e.g., assured by the road authority) and/or *probabilistic* (e.g., estimates based on big data). Communication can be *unidirectional* (the vehicle only receives data) or *bidirectional* (vehicles feed floating car data back). Vehicle information fed back can be neglected, used for offline purposes such as quality management, or directly influence traffic lights (*closed loop/fully cooperative).*

TLA visualization is difficult to categorize. Especially basic technical implementations rely on displaying primarily raw numerical values. One approach to categorizing the user interface could be based on available features such as displaying:

- the next traffic light's *current state*,
- the current traffic light's *residual red-light time* (valuable) or residual green time (not recommended), and
- *recommendations for approaching* the traffic light.

Another category is based on whether or not the TLA incorporates current information into calculations for the *next traffic light* or for *multiple* subsequent traffic lights along the route. TLAs can be also divided into those providing *single-way* visualization (e.g., when the route is known from a navigation system or turn indicators) or *multiple-way* visualization (e.g., also showing left- and right-turn lights). The information can be displayed g*enerally for all drivers* as is the case with dynamic signs on the road side or countdowns, or only for an *individual inside a car*. The TLA can be unaware of context, or partly or fully *context aware* (e.g., awareness of waiting queues, speed limits, heavy rain/snow, construction sites, traffic jam, and so forth).

Incorporating test subjects is essential in the field of human factors engineering. Here, traffic-light experiments can be divided into those involving *driving simulators*, those conducted on *test tracks*, and those taking place in *real traffic*. The TLA's intended place of use (*urban* or *rural*) is also an important categorization criterion.

While DSRC has some limitations such as range, it could represent a viable solution within an urban environment. However, traffic centers having information about dozens or hundreds of traffic lights are often established in cities, which might render Internet connection a better alternative. In urban areas, the distance between traffic lights is typically small, traffic density high during the daytime, and the speed range for adaptations narrow. In addition, there are frequent, vulnerable road users who should receive more attention than the TLA from a driver. This situation could impede a driver's productive *manual* use of the TLA in inner cities; *automated* actions of the car or drive train could be a solution. In urban areas, the driver is also frequently able to observe unaided the behavior of the next one or two traffic lights. Some drivers receive additional clues, such as pedestrian or bicycle lights, or become accustomed to the phase sequences.

The traffic lights on rural roads, which feature large separations, typically lower traffic density, greater speed range, and fewer or no vulnerable road users, would be a suitable use case, but they are usually operated autonomously without communication connections. The organizational and financial effort to reliably get information to or from a rural traffic light is disproportionately high. Many local and regional authorities administer these "outback" lights in Germany. Even when they want to cooperate, the lights are produced by different manufactures and could require (expensive) proprietary upgrades to exchange data.

3 Local4Global (L4G)

The Local4Global European research project is based on the concept of technical systems of systems (TSoS) with machine learning capabilities. A TSoS is composed of specific autonomous constituent systems that work in a local environment, optimizing themselves and mutually improving overall performance at the global level. Constituent systems can be of different natures and aim for different objectives. They enjoy the possibility local decision making to a large extent. Remaining decisions are made after all of the participating systems exchange information to learn from each other and improve overall performance.

A generic, integrated, and fully functional methodology or system with the following attributes is needed to develop and extensively test and evaluate technical systems of systems in real life:

- Full autonomy represented by units that react depending only on their local environment to optimize the TSoS's performance emerging at the global level. This attribute renders unnecessary an elaborate, tedious effort to deploy, redesign, or reconfigure the Local4Global system as well as any need for expensive infrastructure.
- A plug-and-play control mechanism.
- Learning, evolving, and self-organizing capabilities.

Advances will lead to a fully-functional, ready-to-use system (Local4Global final product) delivered in the form of embedded, Web-based, plug-and-play software for a generic TSoS that is mountable locally in each constituent system. This system will be deployed, extensively tested, and evaluated in two real-life TSoS use cases: an efficient-building TSoS use case [3] and a traffic TSoS use case. (cf. the http://local4global-fp7.eu project page)

3.1 Traffic Use Case

In the traffic context, two basic classes of constituent systems are suggested: dynamically signalized traffic junctions and connected vehicles with speed control capabilities. Local traffic-signal control is based on a variant of the max pressure algorithm, while local control of vehicle speed relies on the resulting signaling in-formation. Both controls receive a correction from a global optimizer in a bigger, daily, common control cycle. Figure 1 illustrates the test bed implementation.

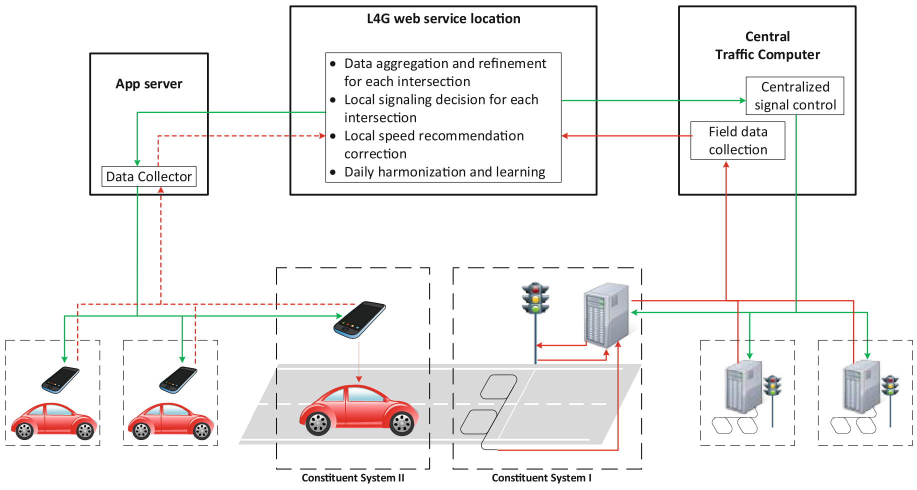

L4G web service location
Data aggregation and refinement for each intersection
Local signaling decision for each intersection
Local speed recommendation correction
Daily harmonization and learning
App server
Data Collector
Central Traffic Computer
Centralized signal control
Field data collection
Constituent System II
Constituent System I

Fig. 1. The real-world implementation's architecture (app server, L4G server, traffic center)

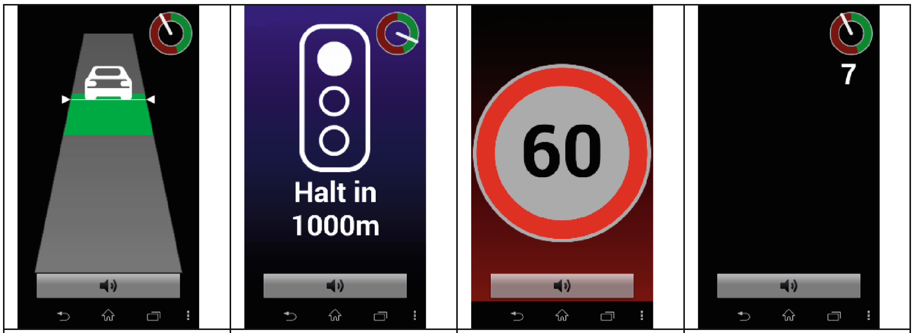
Occupancy and traffic flow data from constituent class I systems (signal control) are gathered and sent to the road authority's central traffic computer. Collected data is subsequently forwarded to the L4G Web service location where data from signal controllers is aggregated and refined. After calculation of new signal plans using a distributed signal-control strategy and optimizer parameters, plans are sent back to the central traffic computer and from there to the signal controllers. The same signal plans are used together with position and speed data collected from mobile telephones to calculate speed recommendations for the constituent class II systems (cooperative vehicles). Data exchange is effected through an application server, where position and speed data are collected together with suggested speed recommendations. To improve the estimation accuracy of speed recommendations, an additional algorithm is integrated for dynamic queue-length estimation.

A microscopic simulation [2] showed that the algorithm effectively adapts automatically to the situation, achieving better performance during evening traffic peak relative to static signaling using an off-peak signaling configuration.

4 Previous Project: KOLIBRI

[4] gives an overview of six human-factors studies and different traffic engineering results from KOLIBRI. The seventh KOLIBRI experiment was reported in [5]. The last stage in KOLIBRI involved an interface that optimized vehicle speed for the next two traffic lights. In L4G, this is simplified to one traffic light; local optimization is also attempted in L4G. The following figures show the display states and visualizations.

The green carpet (Fig. 2a) on the linear speedometer (bottom: 0 km/h, top: 120 km/h; ego car: current speed) shows the range of speeds for getting to the next light while it is green. It is limited to allowed speeds. The display state in Fig. 2b appears if the driver cannot reach the next traffic light while it is green or has to drive too slowly to do so. If the driver drives more than 10 km/h over the limit, the prevailing speed limit appears (Fig. 2c). Figure 2d shows the remaining red-light time (faded out 3 s to 4 s before 0 s) while the driver waits in front of a red traffic light. The sign in the upper right corner shows the traffic light's current state.



(a) Carpet (b) Stop (c) Speeding (d) Countdown

Fig. 2. Display states — human-machine interface

In KOLIBRI, the interface was tested for suitability (driver distraction) with regard to glance durations, mental workload [6], speeding behavior [7], controllability/detection in case of malfunctions [5], and subjective usability ratings ([5] and [8]). The usability ratings and speeding behavior in particular will be used as references for the L4G on-road human-factors evaluation.

In KOLIBRI, the TLA was a prototypical visualization. In L4G, the planned is to make the TLA publically downloadable for testing on this road section. This evolution from a local interface in supervised subject tests (KOLIBRI) to a full downloadable client-server application (L4G) involves remarkable effort. The requirements and implementations are explained in the next section.

5 Application Requirements and Implementation

The requirements imposed on, and the chosen implementation of, the traffic-light-assistant app and server within Local4Global project are explained in the following section.

5.1 General Considerations

The platform/ecosystem was chosen based on the prevalence of Android in Germany. The percentage for Android phones in use was about 70 % at the start of 2015[1]. The market share for sales was typically above 70 % during 2015[2]. The second alternative would have been Apple's iOS with a 15–20 % market share. Some developers nonetheless prefer the Apple ecosystem despite its market share, because Apple's consumers are typically willing to spend more money for applications, which is not an argument for the scientific experiment. Also, the preparatory work in earlier projects was based on Android and even an early prototype with Adobe Integrated Runtime.

A recommendable alternative for future projects (reimplementation) might to use HTML5 sensor and geolocation access. An HTML5 implementation would make the application independent of operating systems (Web application). For this project, a native Android app (Java) has been implemented.

In this project, there was also a discussion about whether or not all calculations should be executed on a central server with the mobile devices displaying only the interface similar to the approach of a thin-client architecture. We retained the system architecture of the former project in which the mobile devices request information such as signal data from a server and execute their calculations locally in the devices. With the thin client architecture, GPS data from the devices would have been continuously transmitted to the server, processed (hopefully) just in time, and the results transmitted back to the device; but on rural roads some signal drops or small dead zones for transmission are not unlikely with a car having no additional antenna. The chosen (fat-client-like) local architecture is more robust against potential transmission/calculation delay and is resilient to some extend if not every request for the slowly changing signal data is successful. The data is transmitted via Internet connection making the system architecture different from other traffic-light assistants, which use Dedicated Short Range Communications (DSRC).

5.2 User Management

Only registered users who accept the terms and conditions will be able to use the app. When first started, the app asks the user about which email address she or he wants to register. A time-limited confirmation link is send to this email address. At each startup, the app checks the authentication with the locally stored email address. The SIM card's

[1] http://de.statista.com/statistik/daten/studie/170408/umfrage/marktanteile-der-betriebssysteme-fuer-smartphones-in-deutschland/.

[2] http://de.statista.com/statistik/daten/studie/256790/umfrage/marktanteile-von-android-und-ios-am-smartphone-absatz-in-deutschland/.

serial number is also used during the authentication process in addition to the email address. Manually entering a credential is therefore unnecessary. Due to the fact that the SIM serial number is presented via a software interface from Android and Android is open source, this number cannot be treated as completely trusted data (e.g., when self-compiled Android versions or mods are used). But for normal users, email and the SIM serial number are judged to be enough authentication to exclude potential problematic subjects from this specific experiment. This policy has to be reconsidered for a production system.

The terms of use explain the reason for the project and experiment, informs the user about the tracking, and offer hints for safely mounting the device in the car.

During the experiment, a user database enables problems to be traced back (feedback/incidents/questionnaires), excludes persons, or contacts winners of the feedback tombola. After the experiment, the user table is deleted and the logged data (see Sect. 5.3) is processed anonymized.

5.3 Logging

User data (tracking) is logged on the test section with a radius of 20 m, so detecting the direction from which a vehicle entered or left the experiment would be possible. The GPS data rate on mobile devices is typically one Hz. The logged data also include the calculated recommendation/visualization and an indication of which traffic signal data was used on the phone for the calculation. So every minute, 60 of these data points and one configuration data field are stored. The configuration data holds usage/configuration information such as whether the device is in portrait or landscape mode, the app's sound level, app version, and display size. This congregated data are stored to a file. Before being written, the content is encrypted with a random AES 128 key. The AES key itself is encrypted with a RSA 2048 pubic key and also written to the file. This common splitting (AES/RSA) attempts to achieve a performance benefit by doing the major encryption work using a faster symmetric approach (AES) and exchange the key with slower asymmetric cryptography (RSA). The server has the RSA private key to decrypt the encrypted AES key and afterwards the decrypt message using the decrypted AES key. The local files on the phone are deleted after they are transmitted, but in case of transmission problems they are stored temporarily and synchronized later. This local data is protected by the encryption.

The app server (Fig. 1) consists of two servers. One server (master) runs with an Apache Web server; and both servers operate with a MySQL database in master/slave configuration. The MySQL master/slave configuration should help in backup and analysis situations. For backups, the slave database could be simply stopped and the backup extracted. If scripts need heavy database read access, they can simply use the slave without affecting the running data collection (master).

The lessons below were learned from previous projects.

Saving data on the phone with a local timestamp and using this timestamp on the server side as a primary key in a database might be a mistake. This is due to the fact that the user can easily change the local time on the phone (maybe large changes) or a phone network could change it automatically (typically small changes) possibly rendering the timestamp non-unique or not monotonically increasing. Using automatically increasing data fields as primary keys can mitigate the problem. Own Network Time Protocol (NTP) syncing on the phone also helps. Only (multiple row) database inserts are used for the sake of performance. MySQL's *max_allowed_packet* size had to be adjusted in a prior project with bigger datasets. The database tables were set up with instructions to create multiple-column indexes, to later speed up foreseeable analysis requests. All script inputs are seen by default as untrusted or insecure and handled with SQL sanitization.

All connections from the app to the server, between master/slave and interfaces use Transport Layer Security (TLS). The two servers (master/slave) were tested and hardened to get an A rating (status January 2016) on a common security test site[3]. The public certificate for the master server was hardcoded (certificate pinning) into the smartphone app. Therefore, the app only connects, uploads, and accepts data from the known master server.

5.4 Estimated Server Load/Data Volume

As described in the previous section, a data file transmitted to the servers contains 60 GPS data together with information about what signal information has been used, calculation results and displayed information (altogether about 35 data fields), and one status/configuration message about the app's general use such as sound muted and landscape mode.

The data is coded as comma-separated ASCII values. Unpacked, one data file (one per minute) has about 17–19 kilobytes. Packing (gz) typically reduces this to 6–8 kilobytes. Encryption is subsequently applied.

The script receiving this data on the server is assumed to be the bottle neck. It was observed that 0.1 s to 0.2 s is typically needed to handle (decrypt, unpack, parse, store into database, and conduct some housekeeping) the received data (Fujitsu Primergy RX200S8; Intel Xeon E5-2620v2; 16 GB; SATA RAID1 non SSD). This setup will accommodate about 60 s/0.2 s = 300 simultaneous users assuming no parallelization. Traversing the approximately 15 km test section requires about 15 min. Therefore, 1200 users/h in both directions seems possible. The road administration indicated that the project would be supported during off-peak hours. This is typically fewer than 500 vehicles/h in one direction on this track. The relatively small server therefore seems suitable for this experiment even in the unlikely event of a 100 % penetration rate for the app. With very large numbers of users, recurrent http poll requests for new signal data would also need to be taken into account.

It is likely that the server (6 cores/12 threads) can handle even more load than the rough worst case estimation. With further optimizations such as binary data transmission and removing data fields specific to this experiment, potential later production systems would have room for improvement. The registration and authentication procedure during app startup also enables server load to be managed by refusing some users in case problems arise.

[3] https://www.ssllabs.com/ssltest.

5.5 Location-Based Service

For the tests in previous, supervised experiments, it was sufficient for the experimenter to start the app manually while the subject is driving. The version for public download would incorporate an Android service that continuously checks the location, starts the app when the test section is approached, and terminates it sometime after the section is exited.

This is a benefit of the native Android app implementation over a purely Web app (HTML5) approach. For future implementations, a hybrid approach could yield a valuable solution. A native service could check the location and start or terminate the Web application as needed.

The current Android service is only enabled, if the user configures it via checkbox. It starts after the device is booted and checks the location continuously. To save battery power, the check period is adapted based on the distance to the test section. Power-consuming GPS requests are discontinued after several seconds (e.g., indoor), and the coarse phone network location is used instead.

5.6 Time Syncing

The traffic lights use DCF77 for time syncing. In the app, the elapsed real time since the device was booted is used to get a relative time. The app also sends and averages some NTP requests to pin the boot time with an offset to Coordinated Universal Time (UTC). With the current relative time to the boot and the known relation between boot time and UTC, Unix timestamps are calculated app internally.

5.7 Signal Data

The app server constantly polls the L4G server (see Fig. 1) for new data about signal-data programs, queue lengths, and a speed optimization factor from the Local4Global algorithm. The L4G server will not only provide the current signal program for each junction but also the next proposed signal program. To simplify the setup for the test section of (only) seven traffic lights, the app server combines all signal data, queue lengths estimations, and the optimization factor into one (small) single file. A later, large scale production system would probably divide this information based on location. So the app would only requests data that is needed, for example on a specific vehicle's navigation route.

The app constantly polls the app server for new signal data (see Fig. 1). This check is performed via a MD5 hash. If the hash changes, it signals new data is available that are download. The hash is used for an integrity check following download.

If polling fails, for instance due to disconnect or timeout, for some sequential requests, or the last successful transmission is older than a threshold (timestamps), then the app stops showing recommendations and displays an error (fail safe).

5.8 Queue Length/Car Stop

The L4G server runs a queue-length estimator called Transqest, (developed by TRANSVER) to dynamically estimate (scale of signaling cycle time) queue length for each direction at a junction. The estimator inputs the traffic lights' detector to predict queue lengths. The estimations made by Transqest are improved using real positions from participating vehicles communicated through the app server to the L4G server. If a car comes to stop in front of a traffic light, the distance to the stop line is immediately transmitted to the app server. The app server anonymizes this information and forwards an appropriate message to the L4G server. There, the information is used to improve the queue length estimation.

Drivers typically drive near the upper speed recommendation. Therefore, even if no waiting queue is reported or estimated, the green time is nonetheless adjusted locally by several seconds (e.g., about 5 s) on the smartphone. This is done for safety, comfort, and in case (an unpredicted) car is waiting. Otherwise a very confident user could approach the stop line with remarkable speed in some cases exactly when the light (hopefully) switches to green. This evokes stress and could be critical.

These adjustments are only incorporated for the speed-recommendation/calculation (carpet visualization); the graphical visualization of the current traffic lights state is not adjusted.

5.9 Diverse Helpers/Functions

The open source tool LimeSurvey is used for user **feedback questionnaires** and incident reports. The opportunity to win money will be offered to motivate users to participate in recurring surveys.

Different things are checked when the apps starts before it displays recommendations. First, the app requests a minimum required version number from the server and compares it with the app's own version. With this **version number check**, outdated versions can be invalidated and redirected to the app's store for download. Whether locally stored route information such as test section GPS track, stop lines, and speed limits are the same on the app server and locally on the phone is also checked (MD5 hashes). On startup, the app also only proceeds if an **NTP sync** was successful. If something fails on startup, the app makes recurrent trials after some time.

With a hidden feature such as multiple taps on a text field, the app can be temporarily switched to a **debug mode** where app-related debug information is transmitted to and stored on the server. This helped a lot during testing and can be useful later for remote troubleshooting, for instance at a help desk via email or phone.

Server scripts have **plausibility checks/asserts** (e.g., input is well formed and has expected length) that automatically log events and send automated admin emails with script information, which also helped during testing. Another purpose is to detect malicious behavior such as a possible intruder playing around with script inputs.

6 Outlook

At the time of writing, the connections between the systems are set up. The complete system can be tested on the road when the connections are established as well as tested, and the road authority grants permission for the L4G system to send signal recommendations (switch signal plan) to a traffic center. When these tests are successful, the last step will be the public experiment lasting for some weeks or hopefully months.

The usage frequency, drop-out rates, subjective feedback (compared to previous, supervised subject tests), and the speeding behavior (compared to previous, supervised subject tests) are of special interest as human factors. The interface itself was thoroughly checked during a previous project with regard to driver distraction (see Sect. 4). Nor did malfunctions inevitably lead to severe incidents for normal drivers. Unexpected events can nevertheless always happen, especially with new systems. The public tests will be clearly marked and communicated as experiments and participants will be registered. System failures can be viewed like those in satnavs, which are unaware of things like changed road speeds, construction sites, new one-way routing, or a ferry-boat connection that is not in place at night.

The drivers need to be careful and attentive (cf. [9]).

The main aim of transmitting information to cars is to increase comfort (hedonic improvement) in a way similar to that of destination and delay displays at train stations, with more relaxed driving and slightly less speeding [7]. There are also indications that using the system can make the ride on a potentially monotonous rural road a little more interesting (cf. [5]). Pragmatic improvements such as potential fuel savings could represent a minor by-product. Optimization of the traffic lights themselves without TLA communication would provide better leverage for achieving pragmatic goals. Improvements would be available to all road users this way.

References

1. Krause, M., Bengler, K.: KOLIBRI – Ampelassistenz für die Landstraße auf einem Smartphone. Zeitschrift für Verkehrssicherheit **60**(3), 135–141 (2014)
2. Aliubavicius, U., Obermaier, J., Fourati, W., Manolis, D., Michailidis, I., Diakaki, C., Kosmatopoulos, E., Krause, M.: Use of system of systems and decentralized optimization concepts for integrated traffic control via dynamic signalization and embedded speed recommendation (submitted). In: Proceedings of 6th Transport Research Arena, 18–21 April 2016, Warsaw, Poland (2016)
3. Sangi, R., Schild, T., Daum, M., Fütterer, J., Streblow, R., Müller, D., Michailidis, I., Kosmatopoulos, E.: Simulation–based implementation and evaluation of a system of systems optimization algorithm in a building control system. In: MED 2016: The 24th Mediterranean Conference on Control and Automation (submitted), Athens, Greece, 21–24 June 2016
4. Dinkel, A., Krause, M., Bengler, K., Ettinger, R., von Dobschütz, A., Bölling, F.: Cooperative optimization of traffic signal control and driver assistance outside urban areas. In: mobil.TUM 2013 (International Scientific Conference on Mobility and Transport), Munich (2013)
5. Krause, M., Weichelt, S., Bengler, K.: Malfunction of a traffic light assistant application on a smartphone. In: Proceedings of the European Conference on Cognitive Ergonomics (ECCE) 2015. ACM Digital Library (2015)

6. Krause, M., Knott, V., Bengler, K.: Implementing the tactile detection task in a real road experiment to assess a traffic light assistant. In: Miller, L., Culén, A.L. (eds.) ACHI 2015, The Eighth International Conference on Advances in Computer-Human Interactions, 22–27 February, Lisbon, pp. 43–38. IARIA XPS Press (2015). ISSN: 2308-4138, ISBN: 978-1-61208-382-7
7. Krause, M., Yilmaz, L., Bengler, K.: Comparison of real and simulated driving for a static driving simulator. Adv. Hum. Aspects Transp.: Part II **8**, 29 (2014)
8. Krause, M., Bengler, K.: Subjective ratings in an ergonomic engineering process using the example of an in-vehicle information system. In: Kurosu, M. (ed.) HCII/HCI 2013, Part II. LNCS, vol. 8005, pp. 596–605. Springer, Heidelberg (2013)
9. Kanz, C., Marth, C., von Coelln, C.: Liability in the case of co-operative traffic and driver assistance systems. Final report (German), Research Project FE 89.0251/2010. BASt-Beiträge (2012/2013)